Point pattern matching for augmented reality

Liming Yang

UNIVERSITE BRETAGNE LOIRE

Centrale Nantes

Thèse de Doctorat

Liming YANG

Mémoire présenté en vue de l'obtention
du grade de Docteur de l'École Centrale de Nantes
sous le sceau de l'Université Bretagne Loire

École doctorale : Sciences Pour l'Ingénieur, Géosciences, Architecture

Discipline : Aménagement de l'espace, urbanisme
Unité de recherche: Ambiances Architectures Urbanités

Soutenue le 7 décembre 2016

Recalage robuste à base de motifs de points pseudo aléatoires pour la réalité augmentée

JURY

Président :	**Pascal GUITTON**, Professeur des Universités, Université de Bordeaux
Rapporteurs :	**Peter STURM**, Directeur de Recherche, INRIA Grenoble Rhône-Alpes
	Eric MARCHAND, Professeur des Universités, Université de Rennes 1
Examinateurs:	**Gilles SIMON**, Maître de Conférences, Université de Lorraine
Directeur de thèse :	**Guillaume MOREAU**, Professeur des Universités, Ecole Centrale de Nantes
Encadrant de thèse :	**Jean-Marie NORMAND**, Maître de conférences contractuel, Ecole Centrale de Nantes

Abstract

Registration is a very important task in Augmented Reality (AR). It provides the spatial alignment between the real environment and virtual objects. Unlike *tracking* (which relies on previous frame information), *wide baseline localization* finds the correct solution from a wide search space, so as to overcome the initialization or tracking failure problems. Nowadays, various wide baseline localization methods have been applied successfully. But for objects with no or little texture, there is still no promising method. One possible solution is to rely on the geometric information, which sometimes does not vary as much as texture or colour.

This dissertation focuses on new wide baseline localization methods entirely based on geometric information, and more specifically on points. Point patterns can be identified by using geometric relationships between points. Based on those assumptions, this thesis proposes:

- A novel point pattern matching algorithm (RRDM) to robustly and quickly register two 2D point sets under perspective transformation. It is validated by augmenting different kinds of paper maps of the same city with information retrieved from a Geographic Information System (GIS) with the help of road intersections.

- Another novel and general point pattern matching algorithm (LGC). It registers 2D or 3D point patterns under any known transformation type and supports multi-pattern recognitions. It has a linear behavior with respect to the number of points, which allows for real-time tracking. LGC is applied to multi targets tracking/augmentation, as well as to 3D model registration.

- A practical method for projector-camera system calibration by using random dots calibration patterns: the user only needs to manipulate a small calibration board (B4 size) for 30s to get precise and repeatable calibration results. It can be useful for large scale Spatial Augmented Reality (SAR).

Besides, I also developed a method to estimate the rotation axis of a surface of revolution quickly and precisely on 3D data. It is integrated in a novel framework to reconstruct the surface of revolution on dense SLAM in real-time.

Keywords: augmented reality, point pattern matching, registration, texture-less tracking, calibration, projector-camera, augmented map, surface of revolution

Résumé

La Réalité Augmentée (RA) vise à afficher des informations numériques virtuelles sur des images réelles. Pour la RA, le problème du recalage est d'une importance primordiale, puisqu'il consiste à calculer la position et l'orientation de la caméra filmant la scène réelle afin d'aligner de manière correcte les objets virtuels dans le monde réel. Contrairement au tracking qui résout ce problème en utilisant les informations de l'image précédente, la localisation à grande échelle (*wide baseline localization*) calcule la solution en utilisant uniquement les informations présentes dans l'image courante. Il permet ainsi de trouver des solutions initiales au problème de recalage et, n'est pas sujet aux problèmes de *perte de tracking* qui peuvent survenir lorsque deux images successives sont trop différentes l'une de l'autre. Le problème du recalage en RA est relativement bien étudié dans la littérature, mais les méthodes existantes fonctionnent principalement lorsque la scène augmentée présente des textures. Pourtant, pour les objets peu ou pas texturés, il est possible d'utiliser leurs informations géométriques qui représentent des caractéristiques plus stables que les textures et qui sont indépendantes des changements d'éclairage de la scène.

Cette thèse s'attache donc au problème de recalage basé sur des informations géométriques, et plus précisément sur les points. Nous proposons deux nouvelles méthodes de recalage de points (RRDM et LGC) robustes et rapides. LGC est une amélioration de la méthode RRDM et peut mettre en correspondance des ensembles de motifs de points 2D ou 3D subissant une transformation dont le type est connu. LGC présente un comportement linéaire en fonction du nombre de points, ce qui permet un tracking en temps-réel. La pertinence de LGC a été illustrée en développant une application de calibration de système projecteur-caméra dont les résultats sont comparables avec l'état de l'art tout en présentant des avantages pour l'utilisateur en termes de taille de mire de calibration, i.e. une mire de taille B4 seulement est nécessaire quelle que soit le distance de mise au point du projecteur lorsque l'état de l'art utilise des mires de plus de 1.5×2m.

Mots-clés : réalité augmenté, recalage des motifs de points, suivi, peu texturé, calibration, projecteur-caméra, carte augmentée, surface de revolution

Publications

This dissertation is based on the following publications, which have appeared in peer-reviewed conference proceedings:

VRST 2014 Liming Yang, Jean-Marie Normand, and Guillaume Moreau. "Robust random dot markers: towards augmented unprepared maps with pure geographic features." *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*. ACM, 2014. (Details cf. Chapter 3).

MVA 2015 Liming Yang, Jean-Marie Normand, and Guillaume Moreau. "Augmenting off-the-shelf paper maps using intersection detection and geographical information systems." *IEEE International Conference on Machine Vision Applications (MVA)*, 2015 14th IAPR. 2015. (Details cf. Section 3.5).

ISMAR 2015 Liming Yang, Jean-Marie Normand, and Guillaume Moreau. "Local Geometric Consensus: a general purpose point pattern-based tracking algorithm." *IEEE Transactions on Visualization and Computer Graphics (ISMAR 2015)* 21.11 (2015): 1299-1308. (Details cf. Chapter 4).

ISMAR 2016 Liming Yang, Jean-Marie Normand, and Guillaume Moreau. "Practical and precise projector-camera calibration." *In Proceedings of the 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR 2016)*. (Details cf. Chapter 5).

3DV 2016 Liming Yang, Hideaki Uchiyama, Jean-Marie Normand, Guillaume Moreau, Hajime Nagahara and Rin-ichiro Taniguchi. "Real-time surface of revolution reconstruction on dense SLAM." *In IEEE International Conference on 3D Vision (3DV 2016)*. (Details cf. Chapter 6).

Résumé substantiel

Chapitre 1 : Introduction

La Réalité Augmentée (RA) vise à afficher des informations numériques virtuelles sur des images réelles. Pour la RA, le problème du recalage est d'une importance primordiale, puisqu'il consiste à calculer la position et l'orientation de la caméra filmant la scène réelle afin d'aligner de manière correcte les objets virtuels dans le monde réel.

Ce problème de recalage en 3D pour être résolu de plusieurs manières différentes en fonction des dispositifs réalisant la capture de la scène réelle. Pour la RA, la capture de la scène se fait le plus souvent via des caméras et dans ce cas, la solution au problème de recalage 3D revient à déterminer la pose (i.e. la position et l'orientation) de la caméra dans le monde réel à partir des images capturées par cette dernière.

Il y a deux grandes familles qui regroupent les différentes méthodes d'estimation de pose : les méthodes dites *Narrow Baseline Tracking* (NBT) et les méthodes dites *Wide Baseline Localization* (WBL). Les algorithmes NBT utilisent la pose calculée dans l'image précédente et fait l'hypothèse de petits mouvements entre deux images consécutives afin de calculer la pose de l'image courante. Ces méthodes ne sont donc pas utilisables quand la pose de l'image précédente n'existe pas (e.g. initialisation) ou n'a pas été trouvée. A l'inverse, les méthodes WBL ne connaissent pas de telles restrictions. Plusieurs méthodes de WBL existent, par exemple celles basées sur des marqueurs artificiels ou des textures. Pourtant, ces méthodes présentes également des limitations et ne peuvent pas marcher quand les marqueurs ne sont pas acceptés et en même temps il n'y pas assez de textures.

L'idée centrale de notre thèse provient du problème dit des *cartes augmentées*, où l'on souhaite rajouter des informations numériques (issues par exemple d'un Système d'Information Géographique - SIG) sur des cartes en papier non préparées (c'est à dire pas de marqueur). L'idée est de pouvoir augmenter plusieurs cartes dont les couleurs sont différentes de la même ville, à partir du même modèle SIG enregistré. Parce que les couleurs des cartes sont différentes, on ne peut que compter sur des caractéristiques géométriques pour les recaler. Les méthodes existantes basées sur les textures ne peuvent pas résoudre ce problème sans avoir à apprendre au préalable toutes les cartes.

La distribution des intersections des routes d'une ville est, dans le cas général en europe, assez caractéristique de cette dernière. Pour résoudre le problème des cartes augmentées, il est théoriquement possible d'effectuer un recalage en se basant sur cet ensemble de points. De plus, le point est la géométrie la plus basique et peut donc se trouver partout. Par conséquent, le recalage par mise en correspondance d'ensembles de points peut potentiellement servir à la wide baseline localization dans beaucoup de situations.

Chapitre 2 : L'état de l'art

Les méthodes wide baseline localization peuvent être séparées en trois : les approches basées sur des marqueurs artificiels, celles basées sur des textures et celles basées sur des géométries.

Le premier système de suivi de marqueur artificiel en RA a été proposé il y a presque 20 ans [Rekimoto 1998]. Ces marqueurs sont souvent planaires, en noir et blanc, et ils contiennent des motifs prédéfinis, distinguables les uns des autres. Par conséquent, ils sont très faciles à détecter dans des images capturées par caméras. Une fois ces marqueurs détectés, la pose de la caméra peut être calculée de manière simple et efficace. Quoique cette approche soit rapide et robuste, la présence de ces marqueurs un peu partout dans la scène est nécessaire, ce qui est envahissant. De plus, ces méthodes résistent mal à l'occlusion des marqueurs : si une partie du marqueur est cachée alors la détection échoue. Enfin, les marqueurs artificiels doivent occuper une certaine taille dans l'image ce qui rend leur utilisation difficile en extérieur.

Les approches basées textures utilisent les points caractéristiques que l'on appelle aussi points d'intérêts. Lors d'un prétraitement hors-ligne, les points d'intérêts sont extraits dans un modèle (généralement, une image). Un descripteur de grande dimension est créé à partir des textures locales autour de chaque point d'intérêt. Ensuite, lors d'une phase en ligne, des points d'intérêts sont détectés dans l'image courante capturée depuis la caméra, puis sont mis en correspondance avec les points d'intérêts extraits de modèle en fonction de la distance entre leurs descripteurs. Grâce aux correspondances, la pose de caméra peut être trouvée. Bien que ces approches soient implémentées dans beaucoup de SDKs commerciaux, elles ne peuvent résoudre les problèmes similaires à celui des cartes augmentées présenté dans le Chapitre 1.

Les approches basées géométries sont moins matures que les deux autres approches. Peu de méthodes temps-réel existent à cause de la complexité du problème

à résoudre. Ces méthodes essayent d'identifier des géométries dans l'image courante et de les mettre en correspondance avec la géométrie d'un modèle connu que l'on souhaite détecter dans l'image courante. Les géométries les plus utilisées sont les régions, les lignes et les points. Les méthodes basées sur les points ont attiré le plus d'attention car le point est la géométrie la plus répandue et donc la plus facile à détecter. La méthode dite des *Random Dot Markers* (RDM) [Uchiyama 2011b], permet de recaler des ensembles de points aléatoirement distribués en temps-réel. Cette technique a notamment été utilisée par [Uchiyama 2011c] pour réaliser une carte augmentée à partir d'un SIG et en utilisant les intersections routières d'une carte détectées manuellement sur la carte. Toutefois, cette méthode échoue lorsque l'on utilise des intersections de routes détectées automatiquement par un programme. Cet échec est dû à la faible robustesse de la méthode quand il y a du bruit dans la détection. Ce bruit va en effet ajouter ou retirer des points à l'ensemble, ainsi qu'introduire des légères différences aux coordonnées (ce que l'on appelle dans la suite le *jitter*) de chaque points détectés dans l'image courante de la caméra. Ceci va faire échouer la mise en correspondance de cet ensemble de points image avec l'ensemble des intersections issues du SIG. L'un des buts de notre thèse est de résoudre ce problème en proposant un algorithme à la fois efficace et robuste qui permettra de prendre en compte ce bruit issu de la détection automatique des intersections de routes.

Chapitre 3 : Marqueurs consistués des points pseudo aléatoires pour des recalages robustes

L'algorithme "Robust Random Dot Markers" (RRDM) est proposé afin de résoudre le recalage de deux ensembles de points 2D sous une transformation perspective de manière robuste et efficace. Par exemple, si on enregistre un ensemble de points modèle P dans RRDM, et on lui donne un autre ensemble de points Q extrait de la scène (i.e. de l'image courante filmée par une caméra), RRDM va calculer une transformation perspective permettant de transformer l'ensemble de points P en (approximativement) Q.

Dans une première phase, RRDM sépare les deux ensembles de points originaux P et Q à mettre en correspondance (i.e. à recaler) en sous-ensembles locaux. Ceci a pour objectif d'éviter une complexité trop grande lors de la mise en correspondance due à l'explosion combinatoire. Un descripteur robuste s'appelant *ratio de deux surfaces* (pour *Two Surface Ratio* – TSR) est proposé et permet de décrire les positions relatives entre les points dans un sous-ensemble. Le TSR reste stable

même s'il y a du jitter et des points ajoutés ou supprimés aux ensembles. Une fois une mesure de similarité calculée grâce au descripteur TSR, un processus de vote local est réalisé pour chaque couple de sous-ensembles, où l'un des membres du couple est issu de P et l'autre de Q. Ce processus de vote a pour le but de mettre en correspondance des points dans les deux sous-ensembles. Une transformation géométrique locale peut alors être estimé à partir de ces correspondances. RRDM propose pour la première fois d'utiliser un consensus sur des transformations géométriques locales afin d'éliminer les mauvaises correspondances. Ceci nous permet d'identifier les correspondances correctes même si leur nombre est très faible par rapport aux mauvaises.

Des études menées sur des points générés aléatoirement par ordinateur montrent que RRDM est plus robuste que RDM dans les cas où il y a du jitter, lorsque des points sont ajoutés ou perdus, ainsi que lorsque la distorsion perspective est grande. En termes de complexité algorithmique, RRDM a un comportement quadratique par rapport au nombre de points. Pour le problème de carte augmentée, RRDM arrive à recaler les intersections de routes issues d'un SIG et celles détectées automatiquement sur les cartes. Il prouve ainsi que le recalage de points peut être suffisamment robuste et efficace pour effectuer un recalage purement géométrique.

Bien que RRDM résolve le recalage de deux ensembles de points de manière robuste, il présente des défauts non négligeables. En effet, RRDM ne peut pas gérer les cas où il existe plusieurs modèles différents (par exemple rechercher parmi plusieurs ensembles représentant les intersections de routes de plusieurs villes). De part son comportement quadratique, il n'est pas assez rapide pour une application en temps-réel lorsque le nombre de points est plus grand que 100.

Chapitre 4 : Consensus géométrique local

L'algorithme "Consensus Géométrique Local" (LGC) est une suite de la méthode RRDM et peut mettre en correspondance des ensembles de motifs de points 2D ou 3D subissant une transformation dont le type est connu (par exemple, perspective, similitude, etc.). LGC possède un comportement linéaire en fonction du nombre de points, ce qui permet un tracking en temps-réel, même lorsque les ensembles à mettre en correspondance contiennent un nombre élevé de points.

Tout comme RRDM, LGC découpe tout d'abord les ensembles à mettre en correspondance en sous-ensembles locaux. Rappelons que LGC supporte plusieurs ensembles de points modèles que l'on va essayer de détecter dans les images issues de la caméra. L'algorithme LGC contient trois modules : un générateur d'hypothèses,

un validateur d'hypothèses et un raffineur. Étant donnée un sous-ensemble local aléatoirement choisi à partir de l'ensemble des points détecté dans l'image de la caméra, le générateur d'hypothèses estime tout d'abord des correspondances présumées et une transformation locale présumée pour chaque sous-ensemble de points modèles. Ces correspondances et cette transformation locale résumées forment les hypothèse. Pour chaque hypothèse, le validateur applique la transformation présumée aux sous-ensembles de points présents dans le voisinage de cette hypothèse. S'il y a assez de sous-ensembles en consensus avec cette transformation présumée, l'hypothèse est alors considérée comme correcte, c'est-à-dire, elle contient des correspondances correctes. Dans une dernière étape, un raffineur est appliqué avec pour objectif de trouver le plus de correspondances correctes possibles pour estimer une transformation précise.

Des études menées sur des ensembles points générés aléatoirement montrent que LGC est plus robuste que RDM tout en étant aussi efficace, et qu'il est plus efficace que le Hashing Géométrique classique et que RRDM tout en étant aussi voire plus robuste. Nous montrons également que LGC est capable de détecter et de suivre divers objets avec ou sans texture en temps-réel, en augmentant par exemple des croquis d'ingénierie avec leurs modèles 3D.

Chapitre 5 : Calibration d'un système de projecteur-caméra

En combinant une caméra et un projecteur reliés de manière rigide ou non, un système projecteur-caméra (ProCam) permet de projeter des informations numériques directement sur des surfaces ou des objets réels (*Spatial Augmented Reality* - SAR). La calibration de système ProCam consiste à déterminer les paramètres intrinsèques du projecteur et de la caméra, ainsi que leurs positions relatives (i.e. paramètres extrinsèques). Ces paramètres sont essentiels pour que les informations numériques soient bien projetées là ou elles le doivent sur les objets réels.

Pour les applications avec une grande zone de projection, le projecteur est souvent mis au point à une distance assez grande. Par conséquent, les projections à courte distance deviennent floues car elles sont faites dans une zone de l'espace pour laquelle le projecteur est hors focus. Les méthodes existantes pour calibrer les systèmes ProCam dans de telles situations ont besoin soit qu'un utilisateur manipule une mire de calibration de traille très importante (e.g. 1.5×2m, soit d'utiliser une mire 3D de calibration dont le modèle 3D numérique est connu de manière précise, soit d'attendre quelques minutes pour obtenir les résultats.

Nous proposons une application basée sur l'utilisation de notre algorithme de

LGC qui permet de calibrer un système de ProCam de manière pratique et précise et ce quelle que soit la distance de mise au point du projecteur. L'utilisateur ne manipule qu'une mire de taille B4 (soit 250×353mm) pendant 30s pour obtenir un résultat aussi stable et précis qu'une méthode représentative de l'état de l'art, alors que cette dernière a besoin d'une mire de calibration 22 fois plus grande que la notre. Des expériences montrent que pour une projection a une distance de 4.5m, l'erreur de re-projection est de 4mm environ.

Chapitre 6 : Reconstruction de surface de révolution

La démocratisation des dispositifs permettant à un utilisateur d'acquérir rapidement et simplement un modèle 3D de l'environnement (i.e. scanners 3D, Kinect) a provoqué un regain d'intérêt pour les approches d'extraction automatique de connaissance dans une scène 3D.

La compréhension d'une scène basée sur l'utilisation de données 3D est importante car elle possède de nombreuses applications comme par exemple la préhension robotique, la simplification de modèle et la réalité augmentée. Le but de l'extraction automatique de connaissance dans une scène 3D est dans un premier temps de détecter l'ensemble des formes primitives (par exemple, sphères, cylindres, etc.) qui composent la scène 3D avant d'ensuite tenter d'établir des relations entre elles.

Les formes primitives actuellement détectées automatiquement par les méthodes existantes sont souvent assez limitées, par exemple, planes, sphères, cylindres, etc. Dans ce chapitre, nous nous intéressons à la détection automatique dans une scène 3D des surfaces de révolution (SoR), puisqu'elles englobent une grande variété de formes primitives. L'estimation de l'axe de rotation est une étape essentielle dans la détection d'une surface de révolution. Les méthodes existantes permettant de détecter les SoR dans une scène 3D ne sont pas satisfaisantes car elles ne sont pas assez précises ou bien ne sont pas suffisamment efficaces. Ce chapitre propose une méthode tirant profit de la symétrie des surfaces de révolution, afin de trouver une formulation simple permettant d'estimer l'axe de rotation de manière précise et efficace. Nous illustrons la pertinence de notre méthode en reconstruisant des surfaces de révolution en temps réel dans des données issues d'un SLAM dense.

Nos résultats sur des troncs de cônes synthétiques montrent que les erreurs d'estimations sur les orientations de l'axe de rotation sont inférieures à $0.5°$ et que le processus peut reconstruire différents objets usuels (bouteilles, tasses, mugs, etc.) issus de surfaces de rotations en temps-réel (45fps). Des expériences menées sur des surfaces de révolutions réelles (deux cylindres) montrent que les erreurs d'estimations

des diamètres sont inférieures à 2mm.

Chapitre 7 : Conclusion

Dans cette thèse nous avons proposé des méthodes pour recaler des motifs de points pseudo aléatoires, puis nous avons illustré la pertinence de nos approches en développant des applications de réalité augmentée et de calibration de système ProCam basées sur nos algorithmes de recalage. Nous avons montré que le recalage des motifs de points est une solution viable pour la localisation à grande échelle dans des applications de réalité augmentée. Les contributions de cette thèse sont :

- Une nouvelle méthode de recalage de points robuste et rapide (RRDM) permettant de recaler deux motifs de points en 2D sous une transformation perspective. Nous avons montré que cette méthode permet d'effectuer le recalage des données issues d'un SIG et de les afficher en réalité augmentée sur des cartes en papier dont les couleurs sont différentes et ce sans préparation préalable.
- LGC, une deuxième méthode de recalage de points plus générale que RRDM. L'algorithme LGC permet de mettre en correspondance des ensembles de motifs de points 2D ou 3D subissant une transformation dont uniquement le type est connu. LGC présente un comportement linéaire en fonction du nombre de points, ce qui permet un tracking en temps-réel et le rend donc particulièrement intéressant pour des applications de réalité augmentée.
- Une application de LGC à la calibration de systèmes projecteur-caméra dont les résultats sont comparables avec l'état de l'art tout en présentant des avantages pour l'utilisateur en termes de manipulation de mire de calibration qui est 22 fois plus petite que celle nécessaire avec une méthode existante représentative lorsque la projection s'effectue à de grandes distances ($\geq$ 2.5m).

Dans le futur, diverses améliorations sur des applications développées seront envisagées pour les rendre plus faciles à utiliser. Il est par exemple possible d'optimiser les performances de LGC pour des applications différentes en choisissant de meilleures valeurs de paramètres. Pour ce faire, il sera nécessaire d'étudier attentivement les influences de chaque paramètre sur le comportement global de l'algorithme. De nouvelles applications font aussi partie de nos perspectives de recherche. Nous pensons par exemple à un SLAM monoculaire pour des environnements peu texturés ou bien à une méthode pour recaler des modèles CAO avec ses modèles issus des scans 3D pour lesquels les approches existantes ne sont pas satisfaisantes aujourd'hui. Enfin, afin de proposer une solution plus générale au problème

de localisation à grande échelle, nous souhaitons proposer de nouvelles manières d'utiliser l'ensemble des informations de textures et géométriques, ainsi qu'étudier d'autres méthodes basées sur des géométries plus complexes que des points, e.g. les lignes, les régions, etc.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

Augmented Reality (AR) is a technology which enhances the real-world environment in real-time by adding some extra information, usually generated by computers. It allows people to perceive some information which is hidden or does not exist in the real-world. One of the most commonly accepted definition of AR is given by Ronald Azuma [Azuma 1997, Billinghurst 2015]: an AR system should have three key properties:

1. It combines real and virtual content
2. It is interactive in real-time
3. It is registered in 3D

Azuma's definition is quite general, not specific to visual information. However, since the vision is usually the most important perception system for humans, most AR research focus on augmenting the real-world environment using visual information. Visual AR is also the focus of this dissertation.

AR is closely related to Virtual Reality (VR), both of which create virtual contents to enhance users' perception. The reality-virtuality continuum [Milgram 1995],

as shown in Fig. 1.1, clearly illustrates the difference between AR and VR. The continuum extends from *Real Environment* (pure real-world environment) to *Virtual Environment* (VR). In VR, people are totally immersed into a virtually created world and no real-world object can be perceived. *Mixed Reality* (MR) is defined as the mixture between *Real Environment* and *Virtual Environment*. AR is a part of MR, where only a part of the *Real Environment* is replaced by some virtual contents. In AR, people can interact with both real and virtual objects.

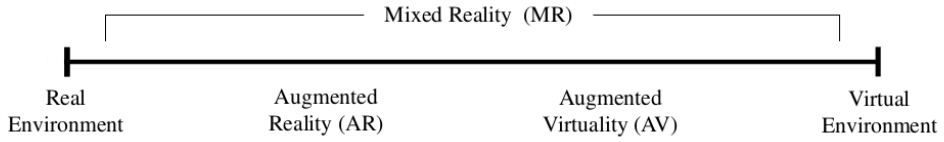


Figure 1.1: The reality-virtuality continuum [Milgram 1995]

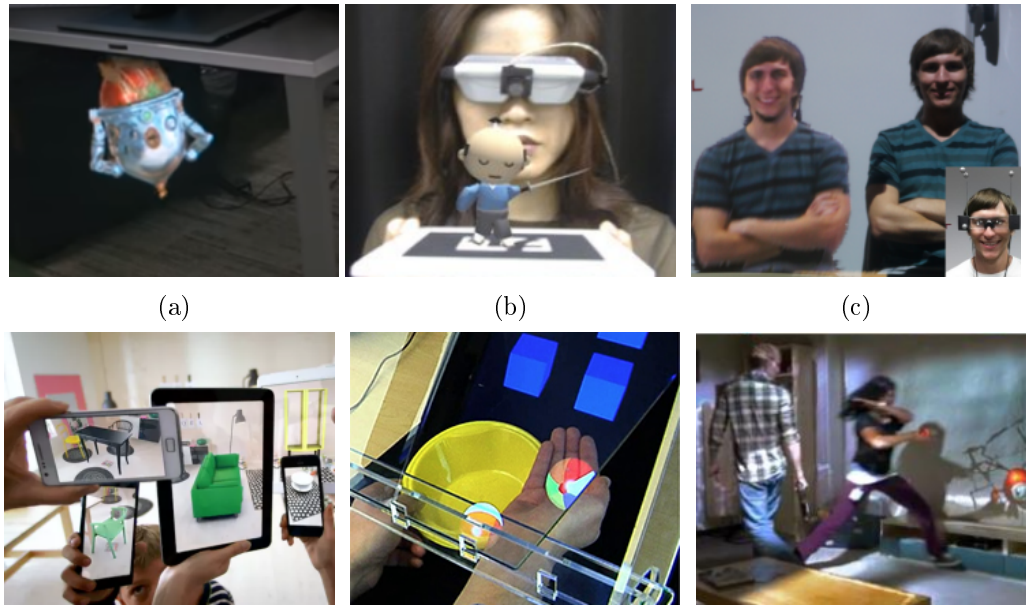
According to Azuma's definition, a basic system of visual AR contains three key components: (1) a 3D registration system, which aligns the virtual environment to the real environment; (2) a rendering system, which combines the virtual environment and the real environment, taking account of occlusion due to different depths, color changes due to lighting conditions, etc.; (3) a display, which shows the rendering result to end users. All the three components should run in real-time (normally $\geq$ 20 frames per second - fps) so that users can notice less artifacts.

Depending on different technologies used to display the augmented contents, AR displays can be classified into three types [Billinghurst 2015]:

1. **Video See-Through (VST)**: VST displays use a camera to capture the real environment. They combine the virtual environment and the real environment numerically and show the final augmented results to users on screens. Users cannot see the real-world directly but only a digital illustration captured by the camera and displayed on screens.

2. **Optical See-Through (OST)**: OST displays use optical systems to combine light from the real environment and light from virtual contents. Users' eyes receive the combined light, so he can see the augmented results.

3. **Spatial Augmented Reality (SAR)**: SAR use projection to directly overlay the virtual environment onto real-world objects. Users directly see both the real environment along with augmented information, without any other device in-between.



(a) (b) (c)

(d) (e) (f)

Figure 1.2: Augmentation shown (a) with OST retina display ([Magic Leap 2015]), (b) with VST Head-mounted display ([Kato 1999]), (c) with OST head-mounted display ([Maimone 2013]), (d) with VST hand-held display ([Ridden 2013]), (e) OST display free of head-wore or hand-held device ([Hilliges 2012]), (f) with SAR ([Jones 2014]).

From another point of view, [Bimber 2005] classifies different AR displays according to the spatial position where the real environment and the virtual environment are mixed, as shown in Fig. 1.3. Different types of displays can be used at different spatial positions. Some examples are presented in Fig. 1.2, such as retina OST (1.2a), head-mounted VST (1.2b) and OST (1.2c), hand-held VST (1.2d), spatial OST (1.2e) and SAR (1.2f).

1.1 Registration

From Azuma's definition, another key feature of AR systems is the 3D registration. It is a common requirement whatever rendering techniques or display devices are used. 3D registration aligns the virtual and the real environments so that virtual objects can be merged into the real-world at their desired positions and orientations. Although 3D registration exists for some VR systems as well, AR systems require much more accurate registration than VR systems. Since virtual objects are overlaid onto the real environment, even a 0.5 degree of arc mis-registration can easily be noticed by users [Azuma 1997].

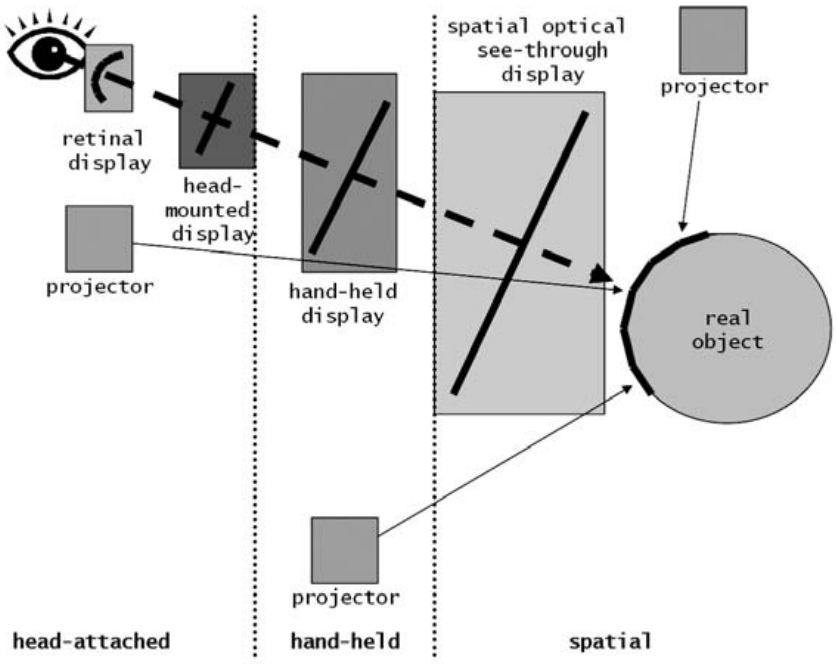


Figure 1.3: AR displays in different position ([Bimber 2005])

When the real environment (or the real target to be augmented) is known, the registration problem is reduced to a localization problem in 3D, namely determining the viewer's pose (i.e. position and orientation) in the real environment coordinate system. Depending on the devices used, localization can be mainly divided into four categories:

1. **Magnetic-based localization** relies on the properties of magnetic fields to calculate the pose of the receiver, which is attached to the viewer.

2. **Vision-based localization** uses information retrieved from captured images by cameras to find the pose of the viewer. Different kinds of cameras may be used, e.g. traditional RGB cameras, depth cameras, etc.

3. **Inertial-based localization** uses Inertial Measurement Unit (IMU), which usually includes accelerometers, gyroscopes and magnetometers. It measures translational movements, the orientation relative to gravity and the velocity of the viewer.

4. **GPS-based localization** relies on satellite navigation to roughly estimate the viewer's geographical location.

5. **Hybrid localization** combines the the advantages of above mentioned approaches in order to give a robust and precise estimation.

The magnetic-based approach needs special preparation on target objects, which increases the workload. Inertial-based and GPS-based approaches can only give a rough estimation of the viewer's position, which usually cannot satisfy the precision requirements of AR systems. The vision-based approach has the advantage to be simple and cheap, i.e. using images captured by cameras to estimate the position of cameras with respective to the world. This method is also similar to the humans' visual perception. For these reasons, vision-based localization gets the most attention in the AR community.

1.1.1 Vision-based localization

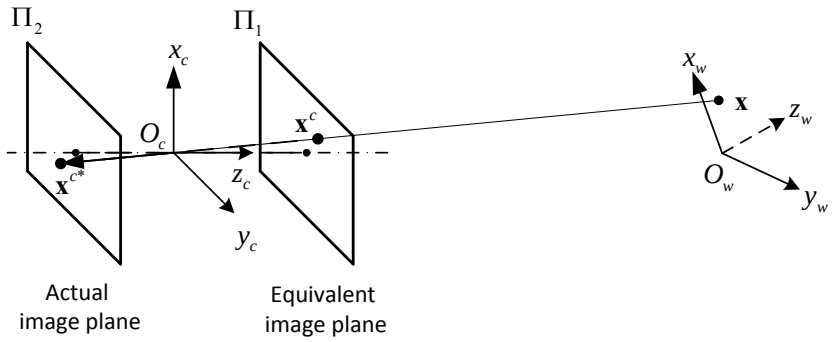


Figure 1.4: Pin-hole camera model

The pinhole camera model is the most commonly used camera model in the computer vision community. Therefore, it is one of the elementary concepts in vision-based localization. The pin-hole camera model is represented in Fig. 1.4, with O_w being the coordinate system of the real environment. O_c is the coordinate system of the camera. Assume that the pinhole camera has zero lens distortion. Light from a world point $\mathbf{x}$ traverses through the camera's optical center and forms an image $\mathbf{x}^{c*}$ on the camera's actual image plane. For the sake of simplicity, the image point $\mathbf{x}^c$ on the camera's equivalent image plane is often used instead. The relationship between $\mathbf{x} = (x, y, z)$ and $\mathbf{x}^c = (x^c, y^c)$ in homogeneous coordinates can be expressed as:

$$\begin{pmatrix} x^c \\ y^c \\ 1 \end{pmatrix} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}] \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{1.1}$$

In (1.1), $\mathbf{K}$ is the camera's intrinsic matrix. It is assumed to be invariant as time and position change. $[\mathbf{R}|\mathbf{t}]$ is the extrinsic matrix, where $\mathbf{R}$ is a rotation matrix and $\mathbf{t}$ is a vector representing the translation. They describe the transformation from the world coordinate system $\mathbf{O}_w$ to the camera coordinate system $\mathbf{O}_c$. As a consequence, performing registration for AR systems with a vision-based approach boils down to finding the camera extrinsic parameter matrix $[\mathbf{R}|\mathbf{t}]$.

In images captured by real cameras, the lens distortion is unavoidable. If $\tilde{\mathbf{x}}^c = (\tilde{x}^c, \tilde{y}^c)$ is the image point distorted by the lens distortions, the distortion-free image point $\mathbf{x}^c = (x^c, y^c)$ can be usually derived by solving equations below: [Zhang 2014, Duane 1971]:

$$\begin{aligned} \tilde{x}^c &= x^c(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x^c y^c + p_2[r^2 + 2(x^c)^2] \\ \tilde{y}^c &= y^c(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1[r^2 + 2(y^c)^2] + 2p_2 x^c y^c \end{aligned} \tag{1.2}$$

where $r^2 = (x^c)^2 + (y^c)^2$, k_1, k_2, k_3 are radial distortion coefficients and p_1, p_2 are tangential distortion coefficients.

Fortunately, since intrinsic matrix and the distortion coefficients can be assumed invariant for a camera as time and position change, cameras in AR systems are often calibrated beforehand. Therefore, localization is to estimate the extrinsic matrix with known the intrinsic matrix $\mathbf{K}$ and lens distortion coefficients.

Most of the time, an AR system seamlessly estimates the location of the viewer (i.e. the camera) since the viewer moves with respect to the environment. It consists of estimating the extrinsic matrix $[\mathbf{R}|\mathbf{t}]$ for each frame in the video sequence captured by the camera. If the viewer's pose in the previous frame is known, it can be used as additional information for estimating the pose of the current frame. The usage of the previous pose to estimate current pose is called *narrow baseline tracking* (also called recursive tracking [Lepetit 2005] or tracking by tracking [Uchiyama 2012]). For the first frame of the input video sequence, or if the pose of previous frame is not found (i.e. the *tracking failure*), the viewer's pose can only be estimated from information in current frame. In such a case, it is called *wide baseline localization* (also called tracking by detection [Lepetit 2005, Uchiyama 2012]).

1.1.2 Narrow baseline tracking

Since the previous camera pose is available, and the current pose is assumed to be changed only little from the previous one, narrow baseline tracking is mostly tackled by optimization, where the previous pose serves as the initial value. Two approaches are commonly used:

1. **Bayesian tracking** recursively estimates the probability density function of camera pose over time using incoming images captured by the camera and a mathematical process model. The two most important applications are Kalman [Gennery 1992] and Particle [Choi 2012] filters.

2. **Local optimization** calculates the camera pose $[\mathbf{R}|\mathbf{t}]$ by minimizing the disparity between the model and the observed data, with the camera pose in previous frame being the initial value. For example, optical flow or template matching minimizes the difference between a template warped by a transformation $T([\mathbf{R}|\mathbf{t}])$ and the input image [Jurie 2001]. Visual Servo System (VSS) tackles the dual problem of 2D visual servoing and minimizes the error between the command and the output of a control system [Comport 2006].

1.1.3 Wide baseline localization

Since no presumed position is available, wide baseline localization consists of finding the camera pose by solving (1.1) in most of the time. As the intrinsic matrix $\mathbf{K}$ and lens distortion coefficients are known, if there are enough world-image points correspondences, (1.1) can be solved and $[\mathbf{R}|\mathbf{t}]$ can be uniquely defined. Depending on the nature of world points' spatial arrangement, there are two main approaches:

1. **Perspective-n-Point (PnP)** is used when world points are not coplanar. When $n \geq 4$, $[\mathbf{R}|\mathbf{t}]$ has a unique solution [Gruen 2002].

2. **Pose from 3D plane** is used when world points are coplanar. The transformation from the plane containing all world points to the image plane can be represented by a homography H. It can be uniquely determined when $n \geq 4$. When both H and $\mathbf{K}$ are known, $[\mathbf{R}|\mathbf{t}]$ can be uniquely determined [Lepetit 2005].

Compared to the "local optimization" in Section 1.1.2, these two approaches aim at finding the global optimal solution. Whichever approach used, the most crucial step in wide baseline localization is to find the correct correspondences between

world points and image points. This is often done by matching points from a previously known *model object* and its image captured by the camera, i.e. points from the *scene object*. In most applications, the number of points to match is much greater than 4. Additionally, there is always noise in image points due to imperfect detections. These constraints make this crucial step difficult to solve.

Some other work treat the wide baseline matching as pose classification plus narrow baseline tracking. Programs learn roughly the pose of the model under different views offline. During online matching, the learnt pose being most similar to the scene is found via classification. Then, the pose is used as the initial value for narrow baseline tracking [Holzer 2009]. This kind of approaches does not need to know point correspondences explicitly.

1.2 Thesis motivation

Although narrow baseline tracking demands less computation efforts and can be relatively easy to solve thanks to information from the previous frame, it suffers from some drawbacks, such as its difficulty to recover from tracking failure and to deal with fast movements (i.e. large movements between two consecutive frames). Moreover, it cannot estimate the initial pose. Wide baseline localization comes with a more expensive computation cost but it does not suffer from any of above drawbacks.

1.2.1 Finding point correspondences

The key of wide baseline localization is to find correspondences between world points and image points. Currently, there are commonly two kinds of approaches to find these correspondences, i.e. marker based approaches and natural feature based approaches.

Fiducial markers, such as ARToolKit Markers [Kato 1999], contain predefined self-distinguishable patterns and are attached to target objects at specific locations. During tracking, they are recognized by the AR system in each frame, thus leading to points correspondences. They can achieve frame-rate wide baseline localization, but they require special preparation on each target, and fiducial markers are intrusive for users.

Nature feature methods are proposed for marker-less wide baseline localization, such as SIFT [Lowe 2004], SURF [Bay 2008] or BRIEF [Calonder 2012]. These methods rely on local textures to distinguish different interest points to establish

point correspondences. Recently, they become a standard approach for wide baseline localization in many cases.

1.2.2 Problem risen from augmented maps

The motivation of this dissertation comes from the idea of *Augmented Maps*. Traditional paper map offers lots of advantages: users can have natural interactions with them such as adding annotations or drawing trajectories. It is also easier for several users to collaborate on a common paper map which facilitates communication. However, paper maps remain static: their content cannot be changed according to users' preferences, no computation result (such as flow simulation or shortest path) can be added without adding layers or reprinting the map. These drawbacks can be removed if paper maps were augmented with dynamic digital information. Since most information of cities is now recorded in Geographic Information Systems (GIS), it is a natural way to base augmented maps on GIS.

(a) (b) (c)

Figure 1.5: Edinburgh: (a)-(b) paper maps of different styles. (c) road network extracted from GIS.

Relying on existing methods, one possible approach is to associate natural features of target paper map with their geo-coordinates manually, then the map can be tracked by traditional natural feature methods. But manual registrations are inconvenient. Especially when working with different types of maps of the same city (cf. Fig. 5.3), manual registrations should be done for each type of map. Therefore, an automatic geo-referencing of paper map with GIS is preferable.

Contents on paper maps and information stored in GIS are very different. For example, Fig.1.5a-1.5b show two paper maps of different styles of Edinburgh, which are raster images. Fig.1.5c is an illustration of the road network of the same area retrieved from a GIS, where raw data are vectorial (e.g. lines, points, polygons, etc.). Natural feature methods are not able to establish correspondences between these two sources, since the nature of data is completely different.

In fact, the only common features between a paper map and the data from GIS

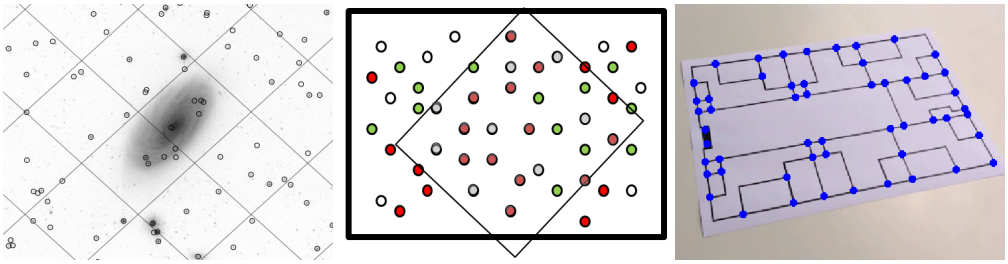
of the same area are geometric quantities, such as road networks. Uchiyama et al. have successfully registered paper maps with a GIS by using a point pattern matching algorithm (Random Dot Markers - RDM [Uchiyama 2011b]) on road intersections [Uchiyama 2011c]. However, they need to explicitly mark the road intersections manually so that they can be easily detected. In order to augment natural, unprepared maps, it is possible to use algorithms, such as [Callier 2011], to automatically extract road intersections instead of performing a manual detection. Unfortunately, such algorithms are prone to massive under and over detection as well as approximation errors of positions of the road intersections. As a result, it is impossible for the RDM method to compute a correct matching with these automatically detected road intersections.

Thus, automatic geo-referencing of paper map with GIS still remains unsolved. More generally speaking, the problem of wide baseline localization when only geometric quantities are available, without any marker or texture, is still not fully solved.

1.2.3 Point pattern matching and its difficulties

Point is the most basic geometric element. In some cases, only points are available, such as a starry sky (cf. Fig. 1.6a) and bead coding used in massive simultaneous assays in drug screening and drug discovery (cf. Fig. 1.6b). In some cases, objects may not have much texture information but it is easy to retrieve lots of points from them, such as engineering drawings (cf. Fig. 1.6c). At last and most important, points can be a very good connection between "images" in different modes, such as road intersections from paper maps and those retrieved from GIS. Therefore, a fast and robust Point Pattern Matching (PPM) method can be a good approach for geometric (or texture-less) wide baseline localization for AR. Although sometimes point may have their own characteristics such as colors, sizes, etc. Pure geometric points with only coordinates are considered here because of their generality.

Point pattern matching, or point set registration, is the process of finding a spatial transformation that aligns two point sets. Let us consider two point sets P and Q, where P is the *model point set* and Q, called the *scene point set*, is an observation of P. PPM aims at determining the pair-wise point correspondences between P and Q, as well as finding the transformation T which brings such correspondences satisfactorily close. PPM has another description known as Largest Common Set (LCS) [Choi 2006]. LCS aims at finding the transformation T which maximizes the cardinality of the subset Q_s from Q and keeps the Hausdorff distance



(a) (b) (c)

Figure 1.6: Some examples of point patterns (a) star matching [Lang 2010], (b) bead coding [Datta 2013], (b) engineering drawing of an apartment.

$dist(T(P), Q_s)$ under a specific small threshold. Due to acquisition noise, points in Q may not originally exist in P (i.e. *over-detection* or *extra points*) and points in P may not appear in Q (i.e. *under-detection* or *missing points*). Moreover, even though there is neither over-detection nor under-detection and the ground truth transformation T is given, the model point set after transformation $T(P)$ may not yet lie exactly on their corresponding points in Q because of noisy point positions. This inaccuracy of point position is called *jitter* afterwards.

The difficulty of PPM is that every point is the same. When considering individual points rather than a group of points, any two points can be potentially matched. This gives a huge number of potential matching candidates. The available information to distinguish different points is only the geometrical distribution of points, i.e. the spatial relationship between points. However, due to the presence of under-detection, over-detection and jitter, the only usable information becomes biased. Given the poorness of the quality of the information, robust matching algorithms are generally time consuming. This is the reason why PPM algorithms in the literature are either fast but not robust against point jitter [Nakai 2006], or robust but not fast enough [Wolfson 1997] for tracking purposes in complex AR applications.

1.3 Thesis statement and contribution

Thesis statement
Wide baseline point pattern matching can be both fast and robust enough for augmented reality applications in texture-less environment.

The main contributions of this dissertation are:

1. A novel point pattern matching algorithm (RRDM) to robustly and quickly

register two 2D point sets under perspective transformation. It is validated by augmenting different kinds of paper maps of the same city with information retrieved from a Geographic Information System (GIS) with the help of road intersections.

2. A general algorithm for pure point pattern matching under any transformation in 2D or 3D in a fast and reliable way. It addresses problems where current methods would fail or require a lot of preprocessing, including but not limited to: model-based augmentation with little texture information (e.g. CAD drawings); augmenting different paper maps with GIS; industrial environments in which placing textures (or classical black and white markers) may be an issue; registering 3D models acquired in different lighting conditions; robust initial pose estimation for edge-based tracking.

3. A practical and precise projector-camera calibration method based on the above algorithm. It can calibrate an out-of-focus projector and gives stable calibration results as well as small Root-Mean-Squared errors (RMSE). To calibrate projectors with large focus distances, traditional methods either require users to manipulate a huge calibration board or to have a precise 3D model. The proposed method only needs a B4 (250×353mm) size calibration board to achieve comparable calibration results.

1.4 Thesis outline

The rest of the dissertation is organized as follows. Chapter 2 presents related work on wide baseline localizations. Chapter 3 proposes a robust and fast method for point pattern matching in 2D, which can be used to register unprepared paper maps with GIS data. Chapter 4 proposes a general point pattern matching method in both 2D and 3D. It is more robust, supports multi-model, and has linear time complexity behavior with respect to the number of points potentially to be matched. Chapter 5 show one application of LGC for calibrating defocused projectors. Chapter 6 presents a piece of work on 3D data, i.e. a method to detect and reconstruct surfaces of revolution in a dense-SLAM sequence in real-time. Chapter 7 draws conclusion of the whole dissertation and perspectives for possible future work.

Chapter 2

Related work

Contents

In the past, two surveys on AR have been published in 1997 [Azuma 1997] and 2001 [Azuma 2001]. An analysis of 10 years of ISMAR publications by [Zhou 2008] has shown the important position of registration, i.e. more than 20% of the total papers are related to "tracking". Recently, a survey on pose estimation for augmented reality was published [Marchand 2016].

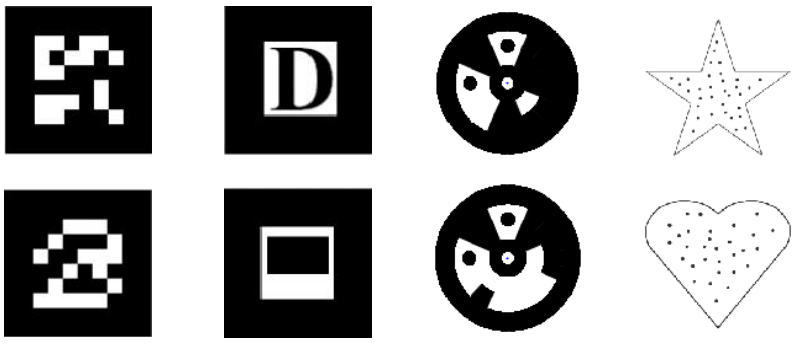
This chapter describes previous work on wide baseline registration methods. It begins with a short summary on fiducial marker based registration methods in Section 2.1, followed by marker-less registration methods based on local textures in Section 2.2. It ends up drawing attention on marker-less registration methods based on geometric information in Section 2.3, which is not totally solved yet but hardly mentioned in latest survey [Marchand 2016].

2.1 Fiducial marker based localization

Fiducial markers are man-made objects, often planar, placed in the field of view of the camera to serve as reference points. Most of these markers contain self-distinguishable patterns, so point correspondences are easy to establish.

Such markers were firstly introduced by [Rekimoto 1998] in AR, followed by many other types of designs. The two most representative types of fiducial markers are square and circular markers. Square markers use four corners of a rectangle as points of interest to estimate a perspective transformation. As a single marker is sufficient to compute the camera pose, they are also denoted as "planar" markers [Lepetit 2005]. While ARTag [Fiala 2005] also use matrix codes as in [Rekimoto 1998] for marker identification, other patterns such as alphabet letters and simple geometric shapes are also used in ARToolKit [Kato 1999]. Regarding circular fiducial markers, at least four of them need to be positioned in a coplanar but non collinear way to compute a unique perspective transformation, thus they are referred to as "point" markers [Lepetit 2005]. Contrasting concentric circles (CCC) [Gatrell 1992] is the main trend of this type of markers for which sub-pixel accuracy can be achieved. [Naimark 2002] adds "data rings" between outer and inner rings to identify different markers.

[Uchiyama 2011b] proposed Random Dot Markers (RDM). They use local arrangements of point patterns, known as Locally Likely Arrangement Hashing (LLAH) [Nakai 2006], to establish point correspondences. Their method proves to be robust to partial occlusions. As this method also can be regarded as geometry based registration, it will be detailed in Section 2.3.4.



(a) (b) (c) (d)

Figure 2.1: Examples of fiducial markers: (a) ARTag [Fiala 2005], (b) ARToolKit [Kato 1999], (c) Circular markers [Naimark 2002], (d) RDM [Uchiyama 2011b].

2.2 Texture based localization

Texture based methods usually rely on local features for wide baseline localization. These local features can be interest points [Lowe 1999] or interest regions [Forssén 2007]. A high dimension textual descriptor is constructed for each local feature, and the distance between descriptors is used to measure the similarity between corresponding local features. Then, feature correspondences can be established according to these similarities. Most of commercial SDKs, such as Metaio (bought by Apple in 2015), Vuforia of PTC, provide localization based on this technique.

The whole procedure contains two stages: (1) Offline training stage: features of model objects are extracted and descriptors are calculated. Then a database is created from these descriptors and the positions of features. (2) Online localization stage: for each input image, features of scene objects are extracted and descriptors are calculated. Then these features are matched with model features stored in the database. Finally, points correspondences are established and the camera pose of the current image is estimated.

There are three basic tasks in the whole procedure above: *feature extraction* is dedicated to localizing visual salient features; *feature description* is used to construct the high dimension descriptor from the texture of the local patch around each feature; *feature matching* is used to establish correspondences according to descriptors.

2.2.1 Feature extraction

Interest points include geometric corners as well as the points in an image where the image intensity changes significantly. Usually, we refer to both of them as *corners* [Rosten 2010] in a general sense. Curvature maxima can be extracted to detect geometric corners from the contours of objects [Awrangjeb 2012]. Second-order derivatives or auto-correlation are used to extract general corners, such as Difference of Gaussian (DoG) [Lowe 1999], Harris corner detector [Harris 1988], Shi-Tomasi detector [Shi 1994]. SURF uses the determinant of the Hessian matrix and relies on integral images for speeding up [Bay 2008]. Pixel intensity comparison is also used to detect corners more efficiently (FAST) [Rosten 2006]

Thick line and interest regions can be used as features as well. [Grompone 2010] (LSD), [Wang 2009a] (Line Signature), [Zhang 2013] offer some methods for line segment extraction. [Donoser 2006] gives a good method for extracting interest regions (MSER).

Regarding the wide baseline localization problem, feature repeatability is an important challenge for the extraction method. A feature is said to be repeatable when the same feature is extracted from different images of the same scene, despite perspective distortions or imaging noise. An evaluation of repeatability of different methods on point feature extraction can be found in [Ballesta 2008], which concludes that Harris corner detector [Harris 1988] works the best under different scales and viewpoints.

2.2.2 Feature description

SIFT proposed by David G. Lowe [Lowe 1999] is one of the most famous feature descriptor. After having extracted interest points, the 16×16 neighborhood of each interest point is uniformly divided into 4×4 subregions. For each subregion, the sum of the gradient magnitudes are calculated in 8 directions. The descriptor of the interest point is created by assembling all the sums of the gradient magnitudes in the 8 directions of all the 4×4 subregions, resulting in a 128 dimension vector (cf. Fig. 2.2). The vector is then normalized to achieve illumination invariance.

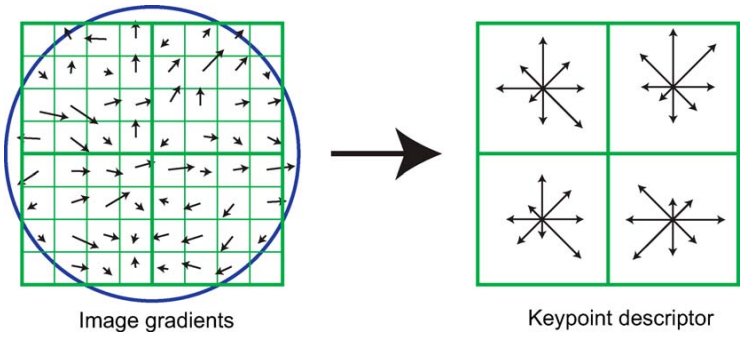


Figure 2.2: SIFT descriptor. Left: Image gradients of 16×16 neighborhood of interest points. Right: Sums of the gradient magnitudes in 8 directions of 4×4 subregions [Lowe 1999].

Following this idea, other kinds of descriptors were then proposed. SURF [Bay 2008] is built by relying on the distribution of first order Haar wavelet responses in x and y directions. BRIEF [Calonder 2012] is created by comparing two sequences of pixels located near the extracted feature, which produces a binary descriptor. These two methods are used for interest point features. For line segment features, [Bay 2005] uses color histograms as descriptor, [Wang 2009b] adopts a SIFT-like approach to create MSLD descriptor. [Forssén 2007] applies SIFT to create an interest region descriptor.

2.2.3 Feature matching

Since descriptors are high dimension vectors, the most common approach is to calculate a distance between different vectors, such as the Euclidean distance [Lowe 1999] or the Hamming distance [Calonder 2012]. For each feature in the scene, one needs to find its corresponding features in the database of model features. This is equivalent to finding the nearest neighbor of the scene feature from all model features in the descriptor space. To this end, either an exhaustive search [Calonder 2012] or approximate kd-trees [Beis 1997] are used. To prevent false matchings, a threshold on the distance value [Bay 2005] or on the distance ratio between the first two nearest neighbors [Lowe 2004] is used. To remove the threshold parameter, one can use PROSAC [Chum 2005], a geometric based approach (cf. Section 2.3.3.1), to help find point correspondences.

From another point of view, feature matching can be regarded as a classification problem given feature descriptors. The view set of a model feature under different transformations defines a class. Classes are trained offline and scene features are classified online. Two features belonging to the same class establish a matching. Random trees [Amit 1997] and random ferns [Ozuysal 2010] have been used and provide good results.

2.3 Geometry based localization

Different from texture based methods, geometry based localization does not rely on any photometric information to establish feature correspondences, or to directly find the camera pose. Since no color information is used, this kind of approach can register information in different representations. For example, the same object being painted in different colors can be aligned with its CAD model with no previously defined color; raster paper maps can be registered with information from GIS; an infrared image of a scene can be aligned with its visual appearance, etc. Due to the lack of discriminative information, methods based on geometric information are often more computational expensive and thus only few work have been proposed for real-time applications, e.g. AR. Therefore, the following of this dissertation is not limited to real-time methods in order to give a wider view in this section.

In 2D images, geometric quantities are points, edges, regions (or shapes), etc. Methods are divided into region based approaches in Section 2.3.1, line segment based approaches in Section 2.3.2 and point based approaches in Section 2.3.3. More specifically, I introduce and discuss in detail *Random Dot Markers*, a point based

approach in Section 2.3.4 as it consists of the background of this dissertation.

2.3.1 Region based approaches

A 3D object usually forms a region (or a shape) in a 2D image. Considering the fact that regions are the most complex geometric structures in 2D images, one can obtain more information than edges or points to perform registration. As a consequence, many texture-less registration methods rely on the shape or the contour of regions.

To register planar objects, [Goshtasby 1986] uses regions' mass centers as control points and applies a point pattern matching method to solve the correspondence problem. [Flusser 1994] creates a 4 dimensional affine invariant descriptor from each region. Each descriptor is associated with its homologous mass center, which facilitates the point pattern matching problem. [Martedi 2013] use joints of line segments extracted from the region's contour as control points. Support Vector Shape (SVS) [Van Nguyen 2013] is a descriptor of shape constructed using Support Vector Machine (SVM) [Cortes 1995], where the resulting support vectors are control points near the contour of the shape. [Campbell 2015] by using control points of SVS boils down planar shape registration to point pattern matching. [Holzer 2009] virtually generates different views of the model regions, then creates Distance Transform (DT) [Rosenfeld 1968] from contours of each virtual view. These distance transforms are then used to train random ferns. The pose estimation problem is solved by online classification. The method requires the target to contain several regions and can achieve near real-time matching. [Rothwell 1992] proposes a method to find four perspective invariant points on a general continuous concave curve. The method is applied by [Hagbi 2009] to calculate a perspective invariant signature for each concave curve in order to distinguish different concave shapes. It can estimate the perspective transformation with only one concave shape. [Donoser 2011] combines the method of [Holzer 2009] and the one of [Hagbi 2009] to work with at least one concave shape. They achieve real-time matching.

To register 3D objects, a common approach is adopted by most researchers. First, 2D views of the 3D model object are virtually generated from a set of possible viewing angles and stored in a database with their corresponding poses in an offline phase. Then during online matching, the best virtual image with respect to the scene image is found by contour matching. At last, the pose corresponding to the matched model image is used as an initial value and refined by a recursive tracking process. [Choi 2012] uses chamfer matching for contour matching and particle filter for pose refinement. [Reinbacher 2010] proposes to use a hierarchical structure to

organize the virtual views in order to speed up the matching. [Holzer 2009] treats the best contour matching problem as a classification problem. He uses the distance transforms of all virtual views to train ferns and the scene image is classified later on. As contour matching is a very essential step, some researchers aim at developing more efficient and robust algorithms. [Damen 2011] discretizes the contour to form edgelets (short straight segments) and creates a constellation of edgelets by casting a ray from the one of the edgelets and following reflections of the ray on other edgelets. The descriptor of this constellation is composed of the parameters (reflecting angles, zig-zag length ratios) of its generating path, which are used for contour matching. BORDER [Chan 2016] is another descriptor for contour matching. It samples the contour to create a sequence of linelets (small line segments of equal length). Angles between linelets are used to calculate the contour's descriptor, which are matched by kd-trees.

2.3.2 Line segment based approaches

Due to noise, occlusion and broken detected lines, the detected endpoints of line segments are hardly reliable [Li 2016]. This effect coupled with over or under detections, makes it difficult to register line segments. [Coiras 2000] uses extrapolation of lines to create triangles. An exhaustive search on triangle matching is adopted to find the final solution. [Guan 2009] also uses lines or extrapolation of lines but converts line matching to point matching by using their intersections. [Wang 2013] transforms lines to points by using their coordinates in the parametric space and thus converts the line matching to a point matching problem. But it does not work for affine transformations. [Long 2014] models each line segment as a 6D point and employs Gaussian Mixture Model and Expectation-Maximization to register the high dimension point set. This method is used to register a map with an aerial image. But it cannot converge when the rotational angular difference between two targets is larger than 40°. [Gros 1998] uses descriptors constructed by neighboring line segments for similarity or affine transformations. It calculates transformations between matched descriptors and finds the clusters of these transformations in parametric space. Line Signature (LS) [Wang 2009a] creates a 11 dimension descriptor for each line segment by using its 5 neighboring segments. It deals with affine transformations and uses a hash table like approach for matching. The BOLD descriptor [Tombari 2013a] is a vector containing two angles between two line segments and the line connecting the middle point of them. BOLD descriptors are matched with the help of kd-trees in order to find correspondences. This method is not affine invariant since angles

change under affine transformation.

2.3.3 Wide baseline point pattern matching

Pure Point Pattern Matching (PPM), i.e. registration of different point sets with only their coordinates, is a common problem in many research field such as image registration, astronomy [Lang 2010] and biochemical tests [Datta 2013]. Major algorithms can be divided into four categories: RANSAC like approaches, geometric invariant approaches, parametric space approaches and optimization approaches.

2.3.3.1 RANSAC like approaches

RANSAC [Fischler 1981] is a random approach for PPM. First one subset of correspondences is randomly chosen among all possible point correspondences in order to estimate a first "hypothesis" of the transformation. Then other correspondences are used to check whether this hypothesis is valid. The hypothesis that gives rise to the highest number of inliers is chosen as final output. RANSAC is an efficient method usually when the inlier ratio (in the sense of correspondence) is higher than 50% [Lowe 2004]. Some improvements on RANSAC were then proposed to cooperate with cases where inlier ratios are lower. BaySAC and SimSAC [Botterill 2009] reduce the probability of correspondences to be selected when they have already led to a false matching in previous hypotheses. NAPSAC [Myatt 2002] is based on the observation that if two correspondences match from two neighboring points in the point set P to two neighboring points in the point set Q, then these two correspondences will probably be correct. It chooses the first correspondence uniformly randomly as in RANSAC. But for other correspondences, it favors the selection of those in the neighborhood of the first correspondence. Nevertheless, these methods can hardly work for pure PPM problems, since the number of all combinations of a model point $p \in P$ and a scene point $q \in Q$ is very large [Denton 2007].

If biased information on point set or correspondences is available other than only point coordinates, other sampling mechanisms are available. With available point correspondence reliability metrics, PROSAC [Chum 2005] draws the correspondences with higher reliability first during the creation of subsets. GroupSAC [Ni 2009] is a better choice when source point sets can be classified into different groups, with the help of additional information from images. [Raguram 2013a] has integrated several other methods mentioned above into a unified package (USAC).

2.3.3.2 Geometric invariant based approaches

Geometric invariant approaches include all methods which exploit inter-point spatial relationships for matching. Geometric Hashing (GH) [Wolfson 1997] is one of the most famous and classical methods in this category. It uses subsets of points to construct coordinate bases so that other points' coordinates under such bases are invariant to a predefined transformation type. In a first offline step, it chooses every possible basis in the model, calculates other points' coordinates in each basis and stores them into a hash table. During the matching, random bases are iteratively selected in the scene, the coordinates of the remaining points' are expressed in each basis and matched with the model basis in hash table. The pair of bases which receives the highest number of votes gives the final result. GH is robust against extra/missing points and jitter but its major drawback is the huge computational complexity in $O(n^b)$, where $b-1$ is the number of points needed to form a basis [Wolfson 1997]. [Lamdan 1991] and [Rigoutsos 1995] discussed the influence of noise on GH. [Wolfson 1997] gives a good review on classical GH. Instead of using the whole point set for each basis, [Lang 2010] only uses the coordinates of two points under the coordinate system defined by two other points to encode geometric relationship into a "codebook", which reduces the memory cost in case of large amount of points. The proposed "codebook" can be seen as descriptors. [Van Wamelen 2004] proposes to match the nearest neighbors of two points to accelerate the process. While these two pieces of work are restricted to similarity transformations, others use affine invariant geometric descriptors. [Heyl 2013] improves over [Lang 2010] by using surface-ratios to deal with affine transformations and uses kd-trees for descriptor matching. Its time complexity is $O(n^4 \log n)$ with respective to the number of points n, which is not possible to be used for real-time application. LLAH [Nakai 2006] also uses surface-ratios but only relies on neighbor points to create descriptors. A hash table is used for descriptor matching. This method will be explained in detail in Section 2.3.4.

2.3.3.3 Parametric space based approaches

Transformations are uniquely defined by transformation parameters. For example, an affine transformation can be determined by 6 parameters of real value. The parameters of a specific transformation type form a high dimension space. The ground truth transformation corresponds to a point in this parametric space. Some methods aim at finding this parametric point in order to solve the point pattern matching problem. Pose clustering [Olson 1997], also known as Generalized Hough

Transform (GHT) [Ballard 1981], is a representative method in this category. It discretizes the parametric space into bins, enumerates all possible transformations and votes in those bins. The bin receiving the highest number of votes is chosen as the result. However, the enumeration makes these methods computationally intensive. [Huttenlocher 1992] divides the parametric space into subspaces and uses the partial Hausdorff distance to measure the matching score, where a smaller score means a better matching. If the matching score of a subspace is smaller than a threshold, this subspace itself will be divided to smaller subspace for further search. [Mount 1999] proposes to use branch-and-bound breadth-first search for acceleration by introducing lower bounds of subspaces. The parametric space is divided into small cells (i.e. subspaces) and the upper and lower bounds of partial Hausdorff distances are calculated for each small cell. A best value is managed and updated when a smaller upper bound is found. If a cell's lower bound is larger than the best value, the cell is rejected and will not be processed further. This process continues until a satisfactory transformation is found. This method is still not efficient especially when the dimension of the parameter space is high. [Datta 2013] combines geometric invariant approach and parametric space approach. It constructs 4-points affine invariants (4PAI) for each point by using their nearest points. Segment-ratios (which are affine invariants) are created from each 4PAI and are used for 4PAIs matching. The matched 4PAIs then generate clusters in the transformation parameter space to find the affine transformation.

2.3.3.4 Optimization based approaches

For optimization approaches, divergence estimators [Jones 2001] of two point sets are defined as objective functions to be minimized. The most classical method of this group is Iterative Closest Points (ICP) [Besl 1992b]. It assigns to each model point its nearest neighbor in the scene point set with the help of an initial guess of the transformation. Then it estimates an updated transformation with these correspondences. These two steps iterate until a threshold is reached. The process minimizes the sum of correspondence distances and guarantees to find a local minimum. But the method is slow for real-time application when the difference between the poses of two point patterns is big and the basin of convergence of the global minimum is quite narrow. [Fitzgibbon 2003] uses the Levenberg-Marquardt algorithm to minimize the same objective function and updates the intermediate point correspondences for derivative computation as well. It improves the basin of convergence to $\pm 60^\circ$ in terms of rotation in some cases. Instead of assuming one-to-one

correspondence based on the nearest neighbors in each step, robust point matching [Gold 1998] proposes one-to-many relaxations. Each model point is assigned to several scene points with different weights according to the distances between them. These weights are arranged in a correspondence matrix, which collapses to a binary (one-to-one) correspondence matrix during a deterministic annealing.

Researchers also adopted Gaussian mixture models (GMM). Point sets are interpreted as a statistical sample drawing from GMM. This representation naturally takes jitter into account and converts a hard discrete optimization to a relatively easier continuous optimization. [Granger 2002] (EM-ICP) uses model point set to construct Gaussian Mixtures, where each model point serves as the center of a Gaussian kernel. It treats the scene points as samples drawn from model Gaussian Mixture and treats the correspondence matrix as a hidden variable. It applies the Expectation-Maximization (EM) algorithm: the transformation is fixed and correspondences are estimated from expectation during the E-step; then correspondences are fixed and transformation is updated by maximizing a likelihood function during the M-step. Although EM-ICP is more robust than the original ICP, it is about two times slower for the same number of points. [Jian 2011] models both point sets as Gaussian Mixtures and minimizes the L2-distance between the two Gaussian Mixtures. At beginning, the size of the Gaussian kernel is set large, and thus a rough alignment can be found by minimizing the L2-distance. Then, the size of the Gaussian kernels is gradually decreased (i.e. annealing) to achieve a finer alignment. Point correspondences are not found explicitly. This method can be seen as Kernel Correlation (KC) [Tsin 2004] specialized with a Gaussian kernel. [Jian 2011] also points out that the objective function of their method, the original ICP, EM-ICP and KC are all concrete forms of a general divergence family [Jones 2001]. Similarly, the performance of these methods depends highly on the rotation difference between two point patterns.

Besides linear transformations (also called rigid transformation in most work), researchers get more and more interested in non-linear (i.e. non-rigid) transformations, which can be used for perspective transformation for example. [Chui 2000] adds a regularization term in the objective function which limits the magnitude of second order derivatives of the transformation function. [Myronenko 2010] uses motion coherence theory, which imposes the preservation of topological structure on point sets, to regularize the transformation function. But these modifications did not improve the robustness against rotation.

2.3.3.5 Other approaches

[Wang 2012, Caetano 2006] treat point pattern matchings under Euclidean transformations as graph matching problems. As their methods base on the invariance of distances between points, it cannot be generalized to affine or perspective transformations. [Shapiro 1992] proposes the *spectral context*. It calculates the Gaussian distance between every point pair in the same point set and constructs a proximity matrix from these distances. Singular Value Decomposition (SVD) maps each point to an eigenvector of the proximity matrix. In this way, two point sets can be converted to two sets of eigenvectors and the distances between eigenvectors are used for matching. However, this method breaks down if two patterns have different numbers of points. [Tang 2014] uses the same idea but on local point sets to construct descriptors for matching and introduces an "approximate distance order" to deal with noise. Unfortunately, this method takes only similarity transformations into consideration. Evolutionary algorithms [Agrawal 1994, Ezoji 2006, Yin 2012, Zhang 2003] and Particle Swarm Optimizations (PSO) [Yin 2006] have also been used. But they either deal with only similarity transformations, or require certain model points to appear in the scene.

2.3.4 Random Dot Markers

Although there exist numerous approaches for the point pattern matching, algorithms which are efficient enough for real-time wide baseline localization are quite limited. From this point of view, feasible methods are the local geometric invariant based approaches. The most representative method is LLAH [Nakai 2006], which can be applied to achieve real-time tracking of Random Dots Markers (RDM) [Uchiyama 2011b] (cf. Fig. 2.3). It is the starting point of this dissertation and is the closest related work. Therefore, this section provides a detailed explanation of this method and discusses its drawbacks.

2.3.4.1 Locally Likely Arrangement Hashing (LLAH)

LLAH was originally designed as a document retrieval system. The original parameters of LLAH are chosen here for the sake of a simple explanation. LLAH is based on area-ratios, which are affine invariants, to create descriptors. Given four coplanar points A, B, C, D and $S(\cdot)$ being the surface of a triangle, the invariant f is defined as follows:

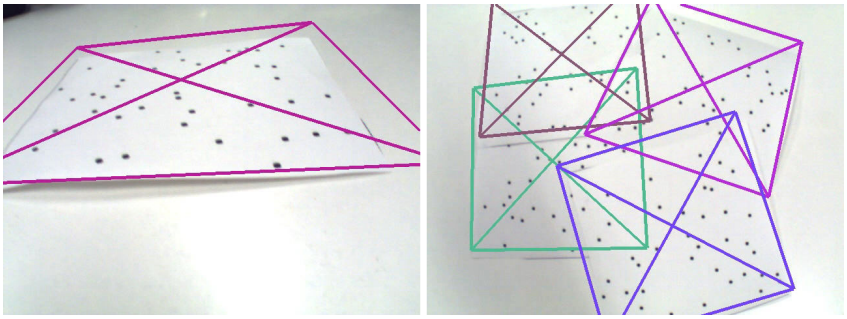


Figure 2.3: Random dot markers [Uchiyama 2011b]

$$f = D\left(\frac{S(\Delta ACD)}{S(\Delta ABC)}\right) \tag{2.1}$$

$D(\cdot)$ is a discretization function which converts real number area ratios into $K = 32$ integers. For each point, the nearest 7 neighboring points are selected to form a neighboring set. All the combinations of 5 points in this neighboring set are used to construct a descriptor. So in total $C_7^5 = 21$ descriptors are created for each point. This design makes the algorithm robust against neighborhood relationship's changes due to over/under detection or jitter.

For each descriptor, the discretized area-ratios of all the combinations of 4 points out of 5 points are used, i.e. 5 ratios. These 5 ratios f_0, .., f_4 are concatenated head-to-tail to form the descriptor, i.e. an integer $d = f_0 f_1 f_2 f_3 f_4$.

The matching process is done in two phases: (1) descriptors of model points are calculated offline and stored in a hash table; (2) during online matching, descriptors of scene points are calculated. Then model points are found in the hash table by matching descriptors. A vote is casted for each model-scene point matching. RANSAC is finally used to find the homography. Authors claim that the time complexity is linear to the number of points [Nakai 2006].

RDM is developed upon LLAH to track random point patterns in real-time. Since LLAH cannot match point pattern under large perspective distortions, RDM additionally manages a dynamic hash table that is updated from the previous frame. As a consequence, the tracking of RDM should be initialized from an approximate front view.

2.3.4.2 Limitations

The efficiency of the algorithm comes from the fact that it uses 5 surface ratios to construct one descriptor, which makes it discriminant. The algorithm is ro-

bust against small noise thanks to the use of several descriptors per point. RDM has successfully been used to augment paper maps with GIS when road intersections are manually extracted [Uchiyama 2009]. But when we use an automatic extraction of road intersections instead (i.e. more and larger detection errors), such as [Callier 2011], it does not work any more.

To summarize its limitations, LLAH/RDM is quite sensitive when noise becomes bigger or the perspective distortion is large. One of the most common noise is the one on points' positions. Due to optical or numerical noise, the detection of a point may be located at several pixels away from its ground-truth position (i.e. jitter). This effect is more obvious when points to be detected are not visually salient, such as road intersections on a map (cf. Fig. 2.5). As to perspective distortions, the position of a point under perspective transformation is slightly different from that under affine transformation if the depth of field is small. This difference has a similar effect as the jitter. Hence if an algorithm can manage well jitters, it is able to deal better with perspective distortions too.

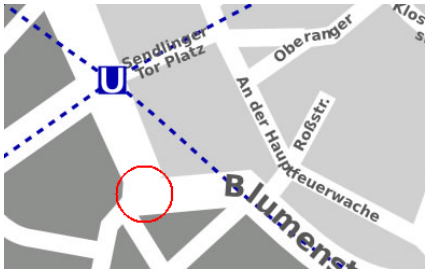


Figure 2.4: An intersection on a paper map is an area (represented by the red circle for example). It is difficult to tell the precise location of the intersection.

A synthetic experiment is performed to study the influence of jitter on LLAH. The model set P contains 40 points randomly distributed over 600×600 pixel squares. The scene set Q is a duplication of P. Then Gaussian noise $\delta \sim N(0, \sigma^2)$ is added on each coordinate component of all points in Q. Fig. 2.5 shows that the number of descriptors which can be correctly matched dramatically decreases with respect to the augmentation of noise on point coordinates. This leads to a total failure of the algorithm when the noise σ becomes larger than 4 pixels.

The reason for the algorithm's sensitivity can be explained as follows: when the position of a point changes slightly, the surface ratio in (2.1) changes but may not influence its discretized value f. But if the noise on points' positions is larger, it is more likely that f changes as well. Since the descriptor is a concatenation of 5 fs, the change of any f_i leads to a different integer, and thus the matching fails.

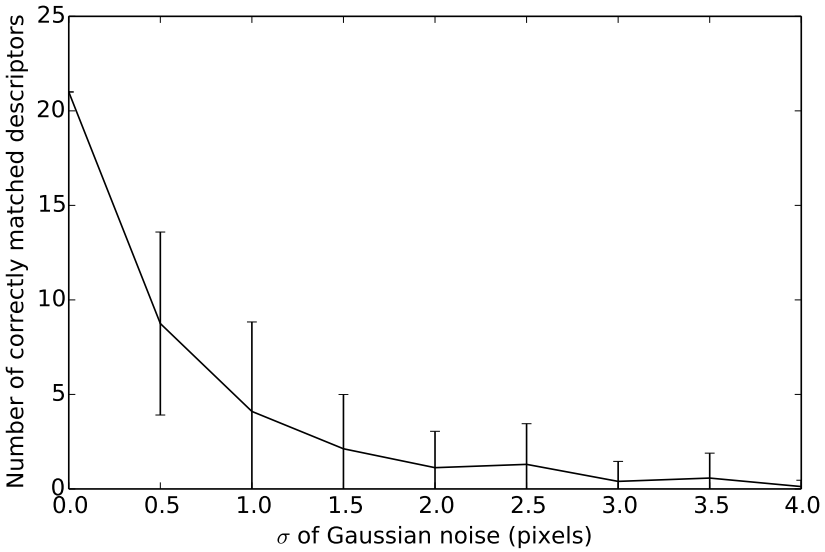


Figure 2.5: The number of correctly matched LLAH descriptors for each point with noise on coordinates. 21 indicates a full match (each point is associated with 21 descriptors).

The discrimination of the descriptor makes it quite sensitive to noise on points' positions. It is difficult to improve the algorithm's performance by simple adjustments because of the following constraint: if one makes the descriptor more robust (by reducing K for example), it will become less discriminant and thus increases the number of false matchings. The latter makes the vote result less reliable, which will make RANSAC work slowly, and maybe even lead to a matching failure.

In a word, the dilemma between robustness and speed of PPM is not solved by LLAH or any other method. One needs to develop new algorithms to achieve better performance.

Chapter 3

Robust random dot markers (RRDM)

Contents

In this chapter, we focus on overcoming the limitations of RDM mentioned in Section 2.3.4.2. By considering PPM between one model point set and one scene point set, we propose a robust matching algorithm which is able to register automatically extracted road intersection from paper maps with GIS data. This work

has been presented at the symposium on *Virtual Reality Software and Technology (VRST)* in 2014.

We reformulate the problem mathematically as follows: let P be a *model point set* in 2D coordinates; H an unknown 2D homography. Let Q be an observed *scene point set*, of which some points belong to $H(P)$ but with small differences in their positions, while others are randomly added noise points (i.e. extra points). The problem is to determine H given P and Q (cf. Fig. 3.1). We assume that **points in P are uniformly randomly distributed**. The challenges of the problem are large point jitter, extra and missing points (cf. Section 1.2.3). Moreover, we have to estimate 8 parameters of the 2D homography H, which is more difficult than 2D similarity (with 4 parameters) and 2D affinity (with 6 parameters).

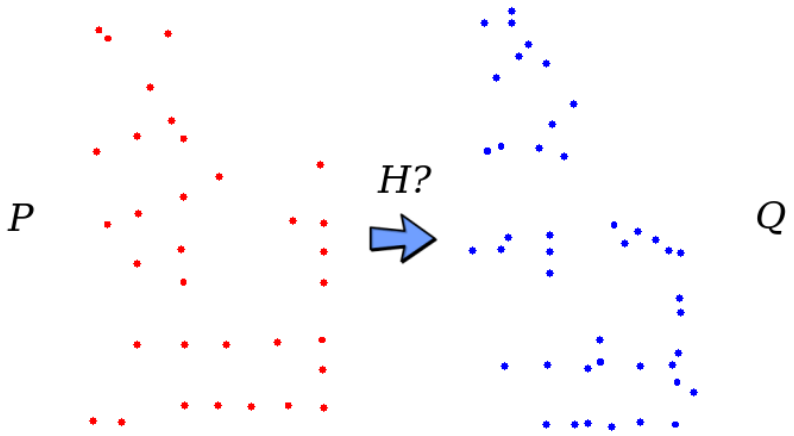


Figure 3.1: P is the model point set in 2D coordinates, Q is the observed scene point set, they are related by a 2D homography H.

3.1 A robust descriptor

We know that (a) pure point pattern matching can only rely on the geometrical pattern itself; and that (b) the presence of extra/missing points and point jitter can significantly alter local geometry of the point pattern. Our task is difficult since we have to take into account both (a) and (b) at the same time. As seen in Section 2.3.4, the LLAH descriptor is an integer composed of five surface ratios computed from seven neighboring points. Any change in those seven neighbors can modify the descriptor, making it very sensitive to noise. In order to solve this problem, we propose a *Two-Surface-Ratios* (TSR) descriptor. It is constructed only by 4 points (a so-called *Quad-Point*, cf. Fig 3.2), so that it is less likely to be influenced by (b).

We choose four points since fewer points cannot provide any affine invariant. In this section, we define the TSR descriptor and show how two jittered Quad-Points can be matched under an affine transformation which lays the groundwork for our new algorithm.

3.1.1 Definition

We call any four-coplanar-point configuration a *Quad-Point*. We do not consider the situation where three points are aligned since it rarely occurs for randomly distributed points. Each Quad-Point is composed of a main point M and three associated points F, R, L. Under such a definition, every Quad-Point can be categorized into one of three possible configurations c. For each configuration, the spatial relationships between these four points, namely M (main), F (face), R (right), L (left), are illustrated in Fig. 3.2. Especially, when $c = 3$, F is chosen so that the surface of triangle ΔMRL is the second smallest among the four triangles. As a consequence, a Quad-Point, denoted as $\mathbb{Q} = \langle M; F, R, L\rangle$ (note that the order of the points is important), is uniquely defined when four coplanar points are given and its main point M chosen (we therefore say that the Quad-Point belongs to M and denote it as $\mathbb{Q}(M)$ when all four points are known).

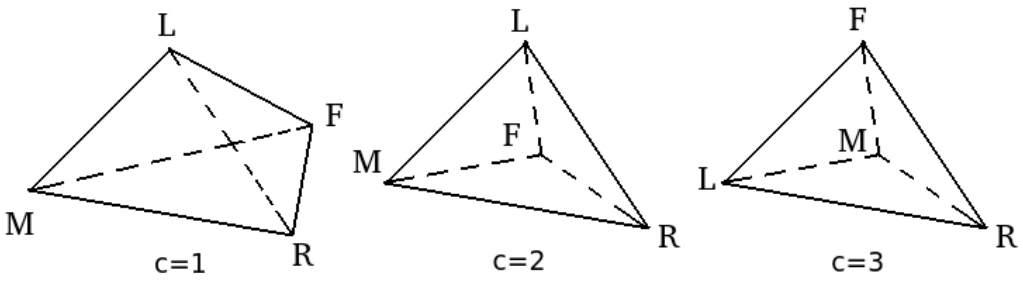


Figure 3.2: Three Quad-Point configurations.

Two Quad-Points $\mathbb{Q} = \langle M; F, R, L\rangle$ and $\mathbb{Q}' = \langle M'; F', R', L'\rangle$ are affinely equivalent if there exists an affine transformation $\mathscr{A}$ between them.

$$\mathbb{Q} \equiv \mathbb{Q}' \Leftrightarrow \exists \mathscr{A} \text{ s.t. } M' = \mathscr{A}(M), F' = \mathscr{A}(F), L' = \mathscr{A}(L), R' = \mathscr{A}(R) \tag{3.1}$$

All Affinely Equivalent Quad-Points (AEQ) can be grouped together. Since a Quad-Point contains exactly two independent invariants [Hartley 2004], each AEQ group has two degrees of freedom.

We propose to rely on two surface ratios along with a configuration specifier to describe each AEQ group. Given its nature, we name our descriptor Two-Surface-

Ratio (TSR). In this context, $\mathbb{Q}(M)$'s TSR descriptor, $\mathbf{t}(\mathbb{Q}(M))$, is defined as follows:

$$\mathbf{t}(\mathbb{Q}(M)) = (X, Y, c) \tag{3.2}$$

$$with \begin{cases} X = \dfrac{S_{\Delta FRL}}{S_a}, Y = \dfrac{S_{\Delta MFL}}{S_a}, & \text{for } c = 1, 2 \quad (3.3) \\ X = \dfrac{S_{\Delta MRL}}{S_a}, Y = \dfrac{S_{\Delta MFL}}{S_a}, & \text{for } c = 3 \quad (3.4) \end{cases}$$

where S_a is the surface formed by the convex hull of $\mathbb{Q}(M)$.

We prove in the following section that two Quad-Points are affinely equivalent if and only if their TSR descriptors are equal. Note that in the absence of point jitter, when two Quad-Points are matched under an affine transformation, they have the same TSR descriptor.

3.1.2 Affine invariance

We want to demonstrate that two Quad-Points are affinely equivalent if and only if their TSR descriptors are equal, which means:

$$\mathbf{t}(\mathbb{Q}) = \mathbf{t}(\mathbb{Q}') \Leftrightarrow \mathbb{Q} \equiv \mathbb{Q}' \tag{3.5}$$

As $\mathbb{Q}$ and $\mathbb{Q}'$ have the same affine invariant, the backward demonstration is direct from (3.1) and (3.2). We focus on the forward demonstration.

We first show that for a Quad-Point $\mathbb{Q}$, the position of F is uniquely defined by the positions of M, L, R and $t(\mathbb{Q}) = (X, Y, c)$.

We use superscript $\cdot^{(i)}$ to represent a point's i-*th* coordinate. Without losing generality, we translate a Quad-Point so that M locates at $(0, 0)$. The 2D plane is divided into seven regions (cf. Fig. 3.3). According to our definition of TSR (Section 3.1.1), F cannot lay in ③ since it would be a "left point" rather than a "face point". For the same reason, regions ④ $\sim$ ⑥ do not fit. So, F can only lay in ① (c=1),② (c=2) or ⑦ (c=3).

For $c = 1$:

$$\begin{aligned} S_a &= \frac{1}{2}\begin{vmatrix} R^{(1)} & F^{(1)} \\ R^{(2)} & F^{(2)} \end{vmatrix} + \frac{1}{2}\begin{vmatrix} F^{(1)} & L^{(1)} \\ F^{(2)} & L^{(2)} \end{vmatrix} \\ S_{FLR} &= \frac{1}{2}\begin{vmatrix} R^{(1)} - L^{(1)} & F^{(1)} - L^{(1)} \\ R^{(2)} - L^{(2)} & F^{(2)} - L^{(2)} \end{vmatrix} \\ S_{MFL} &= \frac{1}{2}\begin{vmatrix} F^{(1)} & L^{(1)} \\ F^{(2)} & L^{(2)} \end{vmatrix} \end{aligned} \tag{3.6}$$

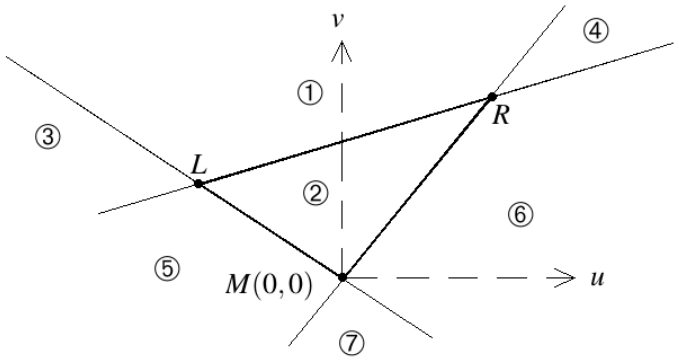


Figure 3.3: Possible regions for F: only feasible regions are ①, ② and ⑦.

With (3.2), we have:

$$\mathbb{Z}\begin{pmatrix} F^{(1)} \\ F^{(2)} \end{pmatrix} = \begin{pmatrix} -R^{(1)}L^{(2)} + R^{(2)}L^{(1)} \\ 0 \end{pmatrix}$$
$$\text{with } \mathbb{Z} = \begin{bmatrix} (1-X)(R^{(2)} - L^{(2)}) & -(1-X)(R^{(1)} - L^{(1)}) \\ -YR^{(2)} - (1-Y)L^{(2)} & YR^{(1)} + (1-Y)L^{(1)} \end{bmatrix} \tag{3.7}$$

This equation gives a unique solution when $|\mathbb{Z}| \neq 0$, meaning that the coordinates of F are uniquely defined. It can be shown that:

$$|\mathbb{Z}| = -(1-X)\begin{vmatrix} R^{(1)} & L^{(1)} \\ R^{(2)} & L^{(2)} \end{vmatrix} = -2(1-X)S_{MLR} \neq 0 \tag{3.8}$$

A similar approach can be applied for $c = 2$ and $c = 3$ for which $|\mathbb{Z}| = 2(1 + Y)S_{MLR}$ and $|\mathbb{Z}| = 2XS_{MLR}$ respectively and where $|\mathbb{Z}|$ cannot be zero.

The forward demonstration is as follows: Assume that $\mathbb{Q} = \langle M; F, R, L\rangle$ and $\mathbb{Q}' = \langle M'; F', R', L'\rangle$ have the same TSR descriptor. We can calculate an affine transformation $\mathscr{A}$ which makes $M' = \mathscr{A}(M)$, $R' = \mathscr{A}(R)$, $L' = \mathscr{A}(L)$. Then let $F^* = \mathscr{A}(F)$, then $\mathbb{Q}^* = \langle M'; F^*, R', L'\rangle$ and $\mathbb{Q}$ are affinely equivalent and thus $\mathbf{t}(\mathbb{Q}^*) = \mathbf{t}(\mathbb{Q}) = \mathbf{t}(\mathbb{Q}')$. Since $\mathbb{Q}'$ and $\mathbb{Q}^*$ have the same TSR descriptor and have three points in common, they are the same Quad-Point, therefore $\mathbb{Q}' = \mathbb{Q}^* \equiv \mathbb{Q}$.

3.1.3 Point jitter and descriptor variance

There are two possible sources for point jitter: (i) distortions from the imaging system and (ii) misdetection of points in the image. The former case can be induced by lens distortions and is both scene and scale independent. In the latter case, point

jitter is scale dependent, becoming smaller when the scene gets further away from the camera. For narrative convenience, in the following we only consider point jitter as arising from the second source.

With the presence of point jitter, the shape of Quad-Points can be changed. The original Quad-Point and its jittered version are generally not affine equivalent, thus they cannot be exactly matched according to (3.1). However, when jitter is not too significant, these two Quad-Points should be "quasi-affine-equivalent". We show that the TSR descriptor can deal with this quasi-affine-equivalence.

We assume that point jitter can be modeled as an additive Gaussian noise to the points' original positions before perspective transformation. Mathematically, each point in the Quad-Point is affected by an additive Gaussian noise $\delta \sim N(0, \sigma^2)$ on two coordinates independently (cf. Fig. 3.4), where σ is small compared to distances between points forming the Quad-Point.

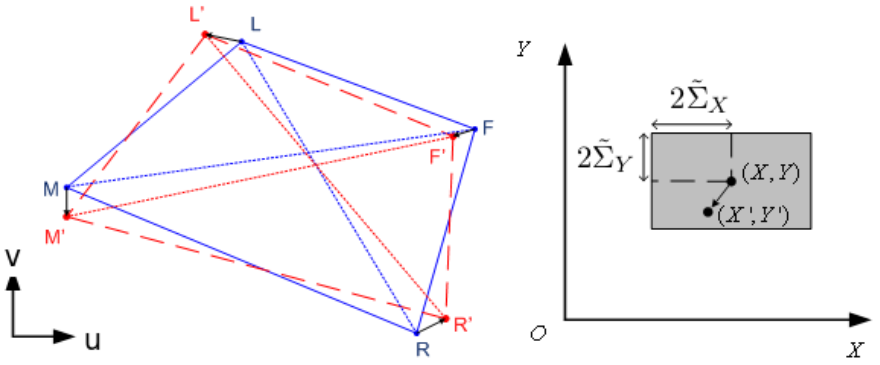


Figure 3.4: Jittered Quad-Point and its TSR descriptor. Left : Original Quad-Point $\mathbb{Q}$ (solid blue line) and jittered Quad-Point $\mathbb{Q}'$ (red dotted line) from a front view. Right : Original (X, Y, c) and jittered (X', Y', c) descriptors in descriptor space. Gray region represents the matching region $R(X, Y, c)$.

Under such assumptions, we can compute the descriptor's variance in first order approximation. Fig. 3.4 illustrates an example of an original and a jittered Quad-Point as well as the related descriptors. For illustration purposes, we compute dX, the difference between the original Quad-Point's TSR descriptor component X and

the jittered one's X', when $c = 1$:

$$\begin{aligned} dX =& X' - X = \frac{S_F dS_M - S_M dS_F}{(S_M + S_F)^2} \sim N(0, \Sigma_X^2) \\ \Sigma_X^2 =& \frac{\sigma^2}{4\,(S_M + S_F)^4} \{ S_M^2 \left(\overline{FR}^2 + \overline{FL}^2 + \overline{RL}^2 \right) \\ & + S_F^2 \left(\overline{MR}^2 + \overline{ML}^2 + \overline{RL}^2 \right) \\ & - S_M S_F \left(\overline{MR}^2 + \overline{ML}^2 + \overline{FR}^2 + \overline{FL}^2 - 2\overline{MF}^2 \right) \} \end{aligned} \tag{3.9}$$

where $S_M = S_{\Delta FRL}$ and $S_F = S_{\Delta MRL}$.

So, when a Quad-Point is slightly perturbed by δ with a small σ, its TSR descriptor will mostly remain unchanged. Thus, we can define two Quad-Points $\mathbb{Q}$ and $\mathbb{Q}'$ such that they are quasi-affinely-equivalent ($\mathbb{Q} \simeq \mathbb{Q}'$) thanks to our TSR descriptor.

$$\mathbb{Q} \simeq \mathbb{Q}' \Leftrightarrow |X - X'| \leq 2\tilde{\Sigma}_X,\ |Y - Y'| \leq 2\tilde{\Sigma}_Y \text{ and } c = c' \tag{3.10}$$

with $t(\mathbb{Q}) = (X, Y, c)$ and $t(\mathbb{Q}') = (X', Y', c')$. Practically, we set $\tilde{\Sigma}_X$ (resp. $\tilde{\Sigma}_Y$) as follows, with Σ_{max} being a parameter of the method:

$$\tilde{\Sigma}_X = \begin{cases} \Sigma_X, & \text{if } \Sigma_X \leq \Sigma_{max} \quad (3.11) \\ \Sigma_{max}, & \text{otherwise} \quad (3.12) \end{cases}$$

For clarity sake, we say that two Quad-Points are quasi-affinely-equivalent when the TSR descriptor of one Quad-Point is inside the matching region of the other. The matching region of $\mathbb{Q}$ is represented by a gray box on the right of Fig. 3.4. In the following, we consider that two Quad-Points are *matched* if they satisfy (3.10).

3.2 Algorithm

Recall that P is the model point set, and it contains the coordinates of m model points. Q is the scene point set which contains the coordinates of n image points that are detected in the image. Each pair (p, q), with $p \in P, q \in Q$ is a tentative correspondence (or in short, a correspondence) that could either be an inlier (i.e. a true correspondence) or an outlier (i.e. a false correspondence).

Both point jitter σ and point pattern density ρ (i.e. the number of points per unit area) play an important role in Quad-Point matching. Given a fixed jitter value σ, the denser the point pattern, the bigger the descriptor variance, cf. (3.9).

We define the *jitter factor* $\eta = \sigma\sqrt{\rho}$ to describe the amount of point jitter (cf. Section 3.1.3).

The main idea of our approach is that even under a large perspective distortion, the transformation between P and Q can be approximated by an affine transformation in a small region. We use local affine information and consistency between neighboring regions to establish point correspondences before estimating an homography.

Our method can be decomposed into two stages. The first one is in charge of registering every model point into hash tables. This stage is performed once offline before online tracking. During the second stage, we retrieve Q from each image, and match it with P by using a Local Voting and Coherency (LVC) strategy.

3.2.1 Offline pre-registration

The offline pre-registration stage is detailed in Alg. 1. P is loaded from a database representing the point pattern. For each point $p \in P$, we find its k-nearest neighbors $N_k(p)$. Each point p is associated with a set of Quad-Points $\mathbb{QS}(p)$, defined as follows:

$$\mathbb{QS}(p) = \{\mathbb{Q}(p) = \langle p; p_1, p_2, p_3 \rangle, \forall p_1, p_2, p_3 \in N_k(p)\} \tag{3.13}$$

The order of p_1, p_2, p_3 is predefined, so each set of Quad-Points $\mathbb{QS}(p)$ contains $\frac{k!}{3!(k-3)!}$ Quad-Points.

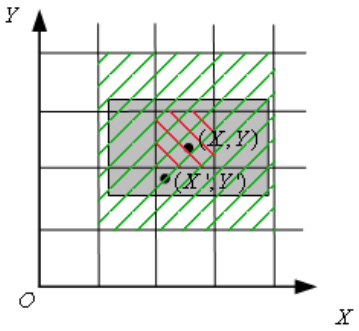


Figure 3.5: Quantization of descriptor value for configuration c. The subregion with red stripes represents $H_{idx}(X, Y, c)$ while subregions with green stripes represent $S_{idx}(X, Y, c)$.

We use three bi-dimensional $W \times W$ (W is an integer parameter of the algorithm) hash tables to register model points so that they can be quickly matched during the online stage with image points according to their descriptors. Each hash

table corresponds to one configuration of Quad-Points illustrated in Fig. 3.2. We determine the set of hash bins $S_{idx} = \{(i, j, c)\}$ into which a Quad-Point $\mathbb{Q}_m$ with TSR descriptor $\mathbf{t}(\mathbb{Q}_m) = (X, Y, c)$ is registered as follows, with $[\cdot]$ represents a floor function:

$$\begin{aligned} [(X - 2\tilde{\Sigma}_X)W] \leq i \leq [(X + 2\tilde{\Sigma}_X)W] + 1, i \in N \\ [(Y - 2\tilde{\Sigma}_Y)W] \leq j \leq [(Y + 2\tilde{\Sigma}_Y)W] + 1, j \in N \end{aligned} \tag{3.14}$$

For clarity sake, imagine that the square $[0, 1]\times[0, 1]$ is uniformly divided into W^2 subregions by vertical and horizontal lines as illustrated in Fig. 3.5. Each subregion represents a hash bin. We register the Quad-Point with $\mathbf{t} = (X, Y, c)$ into the hash bins which intersect the matching region $R(X, Y, c)$.

Algorithm 1 Offline model points pre-registration

```
for p ∈ P do
    Extract ℚ𝕊(p) according to (3.13)
    for ℚ_m ∈ ℚ𝕊(p) do
        Calculate t(ℚ_m) from (3.2)
        Register ℚ_m in all hash bins in S_idx according to (3.14)
```

3.2.2 Local Voting and Coherency

This stage can be decomposed into three phases: (i) pre-alignment by local voting, (ii) rejection by local coherency and (iii) recovery. During the local voting phase, we establish the pre-alignment by computing a score for each correspondence between P and Q. During the rejection phase, we calculate a local affinity for each correspondence using their neighboring points. We discard correspondences for which affinities are not consistent with those of their neighbors. During the recovery step, we try to find inliers that were potentially wrongly rejected in phase (ii).

3.2.2.1 Local Voting

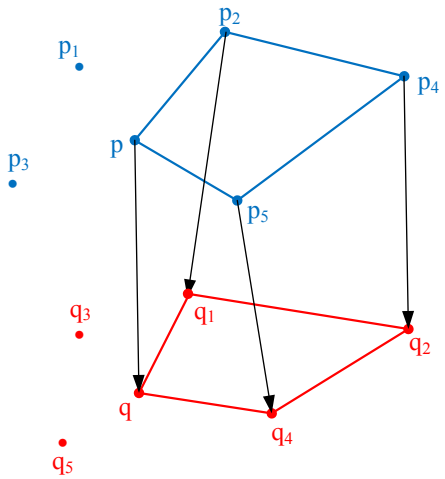
Similar to Section 3.2.1, a set of Quad-Points $\mathbb{QS}(q)$ is generated for each point $q \in Q$. The computational cost is $O(kn \log n + k^3 n)$.

Then for each Quad-Point $\mathbb{Q}_s \in \mathbb{QS}(q)$, a set of model Quad-Points $\mathbb{QM}$ is retrieved from hash bin H_{idx} according to its TSR descriptor (X, Y, c). These matching results are recorded into voting tables. H_{idx} is calculated as follows, where $[\cdot]$ represents a floor function:

$$H_{idx} = ([WX], [WY], c) \tag{3.15}$$

From (3.10), (3.14) and (3.15), we are sure that all model Quad-Points which can be matched with $\mathbb{Q}_s$ are contained in $\mathbb{QM}$.

For each tentative correspondence (p, q), we use a $k \times k$ matrix to record their k-neighbor's corresponding relations, denoted by $V_{p,q}$. These relations are established from two matched Quad-Points as presented in Fig. 3.6. The detailed process is described in Alg. 2. As each matrix only represents the correspondences between two subsets, we call it local voting.



$V_{p,q}$	q_1	q_2	q_3	q_4	q_5
p_1	3		4		
p_2	5+1	4	2	1	
p_3			2		3
p_4		4+1		3	1
p_5	1			4+1	

Figure 3.6: An increment of vote for table of $V_{p,q}$. Left : Neighbor correspondences are established from two matched Quad-Points. Right : Corresponding cells are incremented by one vote in $V_{p,q}$.

Since each scene Quad-Point $\mathbb{Q}_s$ has to be tested against each Quad-Point in $\mathbb{QM}$, the time complexity is $O(k^3nr)$, where the redundancy number r is the average size of $\mathbb{QM}$ across all hash entries.

After local voting, the "quality" (cf. [Chum 2005]) of each tentative correspondence can be represented by the sum of the votes in its voting table. With this measure, a PROSAC-like algorithm can be implemented. However, this approach does not suit our problem as we will show in the experiments of Section 3.4.

Algorithm 2 Local voting procedure

for $q \in Q$ **do**
  Extract $\mathbb{QS}(q)$ according to (3.13)
  for $\mathbb{Q}_s = \langle q; F_s, R_s, L_s \rangle \in \mathbb{QS}(q)$ **do**
    Calculate $\mathbf{t}(\mathbb{Q}_s)$ from (3.2) and $H_{idx}(\mathbf{t}(\mathbb{Q}_s))$ from (3.15)
    Retrieve $\mathbb{QM}$ at H_{idx} from hash table
    for all $\mathbb{Q}_m = \langle p; F_m, R_m, L_m \rangle \in \mathbb{QM}$ **do**
      $V_{p,q}(F_m, F_s)$++, $V_{p,q}(R_m, R_s)$++, $V_{p,q}(L_m, L_s)$++

3.2.2.2 Rejection

This phase aims at rejecting outliers. It begins by clearing votes that are lower or equal to a threshold v_t (which value is discussed in Section 3.3.2):

$$V_{p,q}(\tilde{p},\tilde{q}) = 0, \text{ if } V_{p,q}(\tilde{p},\tilde{q}) \leq v_t \tag{3.16}$$

Let us take one tentative correspondence (p,q) as an example to explain how to find its affinity. (cf. Step 1 of Alg. 3). We first find its local correspondences $s_{p,q}$ by solving a maximum assignment problem. When the number of correspondences $size(s_{p,q})$ is larger than 4, $s_{p,q}$ should contain correspondences from at least two different Quad-Points. This cross validation between Quad-Points helps in reducing false matchings. Thus, we only consider such kind of tentative correspondence and estimate an affinity $\mathscr{A}_{p,q}$ from $s_{p,q}$. Moreover, we assume a statement E_s:

$$E_s : \mathscr{A}_{p,q} \text{ is the correct transformation between local correspondences } s_{p,q}$$

In other words, E_s means that $\tilde{q}$ can be back-projected close enough to its correspondence $\tilde{p}$ in $s_{p,q}$ by $\mathscr{A}_{p,q}^{-1}$. Since $\mathscr{A}_{p,q}^{-1}\tilde{q}-\tilde{p}$ can be regarded as the noise before transformation (i.e. jitter), it follows a Gaussian distribution (cf. Section 3.1.3). As a consequence, χ^2 tests for the sum of squared Gaussian errors can be used to verify E_s.

$$E_s = \sum_{(\tilde{p},\tilde{q})\in s_{p,q}} ||\mathscr{A}_{p,q}^{-1}\tilde{q} - \tilde{p}||^2 < \chi^2_{0.05,size(s_{p,q})-6} \tag{3.17}$$

We use $size(s_{p,q}) - 6$ as the degree of freedom since the estimation of affinity needs 3 points and each point has 2 degrees of freedom. Tentative correspondences which fail the χ^2 check (3.17) will be rejected.

We then group tentative correspondences that have similar affinities into an *affinity group*. Using a Singular Value Decomposition (SVD), we extract two scaling factors $\lambda^{(1)}$, $\lambda^{(2)}$ along two principal axes and one rigid rotation θ from any affinity $\mathscr{A}$. We consider two affinities to be similar ($\mathscr{A}_1 \approx \mathscr{A}_2$) if they satisfy:

$$\begin{aligned} &\Delta\theta \leq \theta_t, \text{ with } \Delta\theta = |\theta_1 - \theta_2| \\ &\alpha_t^{-1} \leq \alpha^{(1)}, \alpha^{(2)} \leq \alpha_t, \text{ with } \alpha^{(i)} = \left|\frac{\lambda_1^{(i)}}{\lambda_2^{(i)}}\right| \end{aligned} \tag{3.18}$$

Note that based on (3.18), θ_t and α_t are two parameters required by our algorithm and will be discussed in Section 3.3.3.

We use the Hungarian algorithm for the maximum assignment problem during Step 1 of Alg. 3. The Hungarian algorithm has a worst-case time complexity of $O(k^3)$. Nevertheless, it should be noted that since there is a large difference in the number of votes between inliers and outliers in $V_{p,q}$ (cf. Section 3.3.2), the algorithm is able to find the optimal solution in a few iterations. Step 1 of Alg. 3 is the most expensive part of the algorithm with a worst time complexity of $O(mnk^3)$. In Step 2, neighboring tentative correspondences having similar affinity are grouped together at first to create small affinity groups, which filters out a large amount of "isolated" tentative correspondences that do not share similar correspondences with their neighbors. Then these small groups are fused if they have similar affinity in order to create bigger affinity groups.

Algorithm 3 Rejection

Step 1. Calculate Affinity
for all $(p, q) \in$ {All tentative correspondences} **do**
s = Solution of maximum assignment problem for $V_{p,q}$
$s_{p,q} = s - \{(\tilde{p}, \tilde{q})\}$, $\forall V_{p,q}(\tilde{p}, \tilde{q}) = 0$
Rejection: mark (p, q) as outlier if $size(s_{p,q}) \leq 4$
Estimate $\mathscr{A}_{p,q}$ using $s_{p,q}$
Rejection: mark (p, q) as outlier if E_s of (3.17) fails

Step 2. Create a set of affinity groups (G)
Group tentative correspondences having similar affinity (3.18) into the same affinity group g. All gs make up a set of affinity groups G. ($g \in G$)

Step 3. Estimate homography for each affinity group (g)
for all $g \in G$ and $size(g) > 4$ **do**
Estimate H_g using RANSAC
Discard $(p, q) \in g$, if $|p - H_g^{-1}(q)| > 2\sigma$

After this procedure, every correspondence in the same affinity group g can be described by the same homography H_g. Since outliers are randomly matched, it is almost impossible to estimate the same homography for several outliers. Therefore, we consider that the largest affinity group contains inliers and call it the *solution list* (denoted as *sl* in Alg. 4). More correspondences will be added to this solution list in the following section.

3.2.2.3 Recovery

We refine the solution list by gradually adding false outliers rejected during Step 2. This increases the number of inliers and results in a better estimation of the

homography between P and Q. The detailed procedure of the recovery phase is shown in Alg. 4.

Algorithm 4 Recovery

```
sl = largest g ∈ G
repeat
  Estimate homography H using sl with least-square method.
  for all (p, q) ∈ sl do
    for all (p̃, q̃) ∈ s_{p,q} do
      if |p̃ − H^{-1}(q̃)| ≤ 2σ then
        sl = sl ∪ {(p̃, q̃)}
until No new element added to sl
Return H
```

3.3 Choice of parameters

The algorithm presented in Section 3.2 relies on several parameters. In this Section, we detail the meaning of each parameter and propose some default values so that one can easily adapt our algorithm to one's own applications and needs.

3.3.1 Robustness of descriptor and k, η, D_{max}

Random addition or removal of points in the close vicinity of a point can deeply modify its k-nearest neighboring points. We call the points not originally present in the neighborhood of a point *false neighbors*. Such false neighbors result in the creation of new Quad-Points in the image which necessarily lead to a reduced number of correct matchings between model points and scene points.

For illustration purposes, let us consider $k = 7$ which leads to the creation of $C_7^3 = 35$ Quad-Points for each point in the pattern. In the presence of 25% additional or missing points (uniformly distributed wrt. the original point pattern) in the image, we can assume that each point will have 2 false neighbors in all its $k = 7$ nearest neighbors. Thus the number of remaining original Quad-Points reduces to $C_5^3 = 10$, meaning that about 10 correct matching are left for each point.

So, robustness with respect to additional or missing points can be improved by increasing k, at the expense of impacting the computational efficiency of the algorithm (cf. Section 3.2.2), since most steps have a computational cost in the order of k^3.

Parameters η and D_{max} are used to manage point jitter. When points in the image are likely to have a high incertitude in their positions, η should be set to a

higher value. When incertitude is so big that (3.12) dominates over (3.11) (resp. for D_Y), a larger D_{max} should be used. Following those guidelines, one can augment the chances to match a *jittered* Quad-Point correctly, thus increasing the robustness of the algorithm against large point jitter. However, increasing η and D_{max} results in enlarging the matching region and in a higher value of the redundancy number r (cf. Section 3.2.2) which in turn impacts the efficiency of the algorithm.

3.3.2 Threshold v_t

Before applying Alg. 3, correspondences that have received less than or equal to v_t votes in $V_{p,q}$ have been rejected. The idea behind this thresholding mechanism is to keep as many inliers as possible while rejecting potential outliers. A brief study of the effectiveness of this threshold is now presented.

Let $Z_i(v)$ and $Z_o(v)$ be two random variables representing the votes respectively received by an inlier and an outlier in voting tables. We find their probability distribution functions $Z_i \sim \phi_i$, $Z_o \sim \phi_o$ experimentally by analyzing 200 randomly distributed points 1000 times with $k = 7$, $D_{max} = 0.2$ and $\eta = 5\%$

Using these parameters values, an important difference in the number of votes received by inliers and outliers is presented in Fig. 3.7. $\varphi_i(1) \approx 0.03\%$ and $\varphi_o(1) \approx 17.3\%$ indicate that 0.03% inliers and 17.3% outliers receive 1 vote. The best choice is $v_t = 8$, since $v_t \geq 9$ will not reject more outliers (in proportion) than inliers.

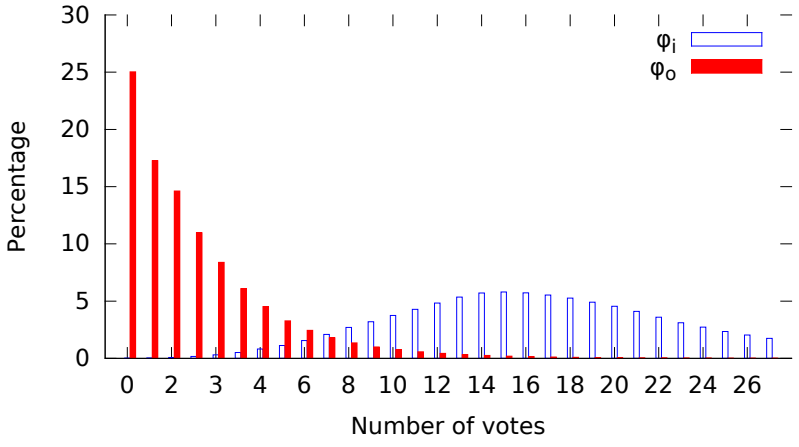


Figure 3.7: Vote probability distribution functions (φ_i,φ_o)

It is worth noting that the choice of v_t has an impact on both the robustness of the algorithm and its speed. Hence, its value should be chosen according to a

trade-off between robustness (lower v_t values) and computational efficiency (higher v_t values).

3.3.3 Influence of point jitter on $\mathscr{A}$ and α_t, θ_t

Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be the affinities of two inliers estimated during step 1 of Alg. 3. Due to the presence of point jitter and perspective distortion, they are usually different even if we have used inliers for their computation. As a consequence, θ_t and α_t have to be carefully chosen in order to decide whether two affinities are similar or not. In the following, we only study the influence of point jitter on the difference between affinities since perspective distortion in a small region can be regarded as a special case of point jitter.

As one can expect, the smaller the distance between points composing a Quad-Point, the more impact point jitter has on the computation of the affinities. Moreover, the fewer correspondences we use to estimate an affinity, the less precise the result. Both observations lead to the fact that the difference between two affinities $\mathscr{A}_1$ and $\mathscr{A}_2$ becomes more important when k gets smaller. In other word, $k = 3$ is probabilistically speaking the worst situation where the difference between $\mathscr{A}_1$ and $\mathscr{A}_2$ might be the greatest.

Experimental results are collected from 1000 runs with 200 points uniformly distributed and $k = 3$. Cumulative distributions for $\alpha^{(1)}$, $\alpha^{(2)}$ and $\Delta\theta$ are illustrated in Fig. 3.8 with $\eta = 5\%$. They are computed from inliers according to (3.18).

With $\alpha_t = 1.3$, $\theta_t = 10°$, $P(|\ln \alpha^{(1)}| \leq \ln \alpha_t) = 91\%$, $P(|\ln \alpha^{(2)}| \leq \ln \alpha_t) = 86.8\%$ and $P(\Delta\theta \leq \theta_t) = 84.7\%$, we show that about $P(|\ln \alpha^{(1)}| \leq \ln \alpha_t) \times P(|\ln \alpha^{(2)}| \leq \ln \alpha_t) \times P(\Delta\theta \leq \theta_t) = 66.9\%$ of inliers affinity pairs are judged similar in the least favourable case.

3.4 Results

In this section, we compare the efficiency and the robustness of our technique (denoted as RRDM), an algorithm combining local voting (cf. Section 3.2.2.1) and USAC [Raguram 2013b] (denoted as USAC) and Random Dot Markers (denoted as RDM). An homography is estimated only if at least 9 inliers are found. This value is given by RDM and we apply this rule to all the three methods for a fair comparison.

For RRDM, default parameters are $W = 100, k = 6, \Sigma_{max} = 0.05, \eta = 3\%, v_t = 4, \alpha_t = 1.3, \theta_t = 10°$. For USAC, all optimization options and PROSAC-based sampling are selected. *inlierThreshold* is set to $2\sqrt{2}\sigma$ since it is the square-

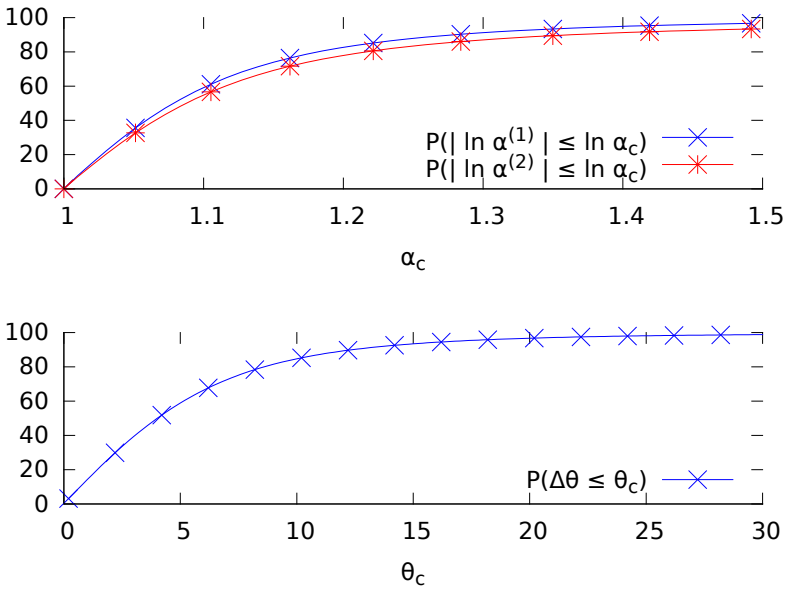


Figure 3.8: Cumulative distribution for $P(|\ln \alpha^{(1)}| \leq \ln \alpha_t), P(|\ln \alpha^{(2)}| \leq \ln \alpha_t)$ (top) and $P(\Delta\theta \leq \theta_t)$ (bottom)

root of symmetric transform error, approximately $\sqrt{2}$ times of squared reprojection error [Hartley 2004]. Other parameters are kept as their default values [Raguram 2013b]. For RDM, only the *tracking by detection* method (i.e. a matching procedure is applied for each image without taking advantage of results obtained in previous frames) of the original Random Dot Markers with default values for parameters [Uchiyama 2011b] is used for a fair comparison. Experiments are done on a PC with an Intel Xeon E3 1240@3.40GHz CPU and 16GB of RAM.

3.4.1 Synthetic images

Synthetic images are used to study the expected matching time in relative to the point quantity, as well as the effects of jitter on each of the three methods. We choose synthetic images because both jitter value and the number of points are difficult to control in real images.

Synthetic images are constructed as a *virtual marker* containing m randomly distributed points in a 1280×720 rectangle. It serves as model point set P. This virtual marker is then contaminated by point jitter which is simulated by additive Gaussian noise, and by $\beta \times m$ randomly generated noisy points. This contaminated virtual marker is placed before a virtual camera so that the angle between the virtual

marker plan and the focal plane of the virtual camera is 30°. Q is the camera image of the point set in the contaminated virtual marker. Remember that Gaussian noise is controlled by η (cf. Section 3.2).

Due to the presence of jitter, an exact estimation is impossible. We use Φ_3 mentioned in [Huynh 2009] to measure if a matching is "precise". With $\mathbf{q_0}$ being the true quaternion of the virtual marker plan and $\mathbf{q}$ the estimated one, a matching is "precise" if:

$$\Phi_3 = \arccos(\|\mathbf{q_0} \cdot \mathbf{q}\|) \leq 1.5° \tag{3.19}$$

In the "Point quantity experiment" (cf. Fig. 3.9), the influence of the amount of points on performance of methods is studied with $\beta = 15\%$ and $\eta = 3\%$. Results are averaged from 1000 experiments. We show that even with a moderate point jitter and a small quantity of extra points, RDM gives a very poor estimation. USAC's estimation is comparable with the one obtained by RRDM but the execution time explodes when m increases due to lower inlier ratios. Our method remains consistently precise starting at $m = 100$ even if RDM remains slightly more efficient than RRDM. This characteristic is very important since for a city like Edinburgh, for example, a map of the city center contains about 200 road intersections. RDM and USAC work best at $m = 100$, so we choose this value for later comparison aiming at a more comparative study.

In the "Jitter experiment" (cf. Fig. 3.10), the LS curve is obtained by using the classical least square estimation **with known** point correspondence and can be considered as an upper bound [Hartley 2004]. Statistics are obtained from 1000 experiments. It can be seen that both RDM and USAC suffer from a very steep performance decrease when η increases.

3.4.2 Real markers

We studied the robustness of the three methods against extra points, missing points and partial occlusion with real markers. Two kinds of equipment are used: a 42 inches (930 × 523mm) LCD screen filmed at 1920 × 1080 by an iPad Air video camera and an A4 paper ($297 \times 210mm$) filmed at 1280×720 with a Logitech C270 camera, as illustrated in Fig. 3.11. A video sequence of 200 to 300 frames is recorded for each case. A visual assessment is provided for all homographies estimated by these experiments. We have to notice that in these real images, no point jitter occurs, which cannot highlight the robustness of our method. We show that even in situations which are in favor of RDM, our method works better as well.

In the "Extra point experiment", $\beta \times m$ extra points are added inside markers to

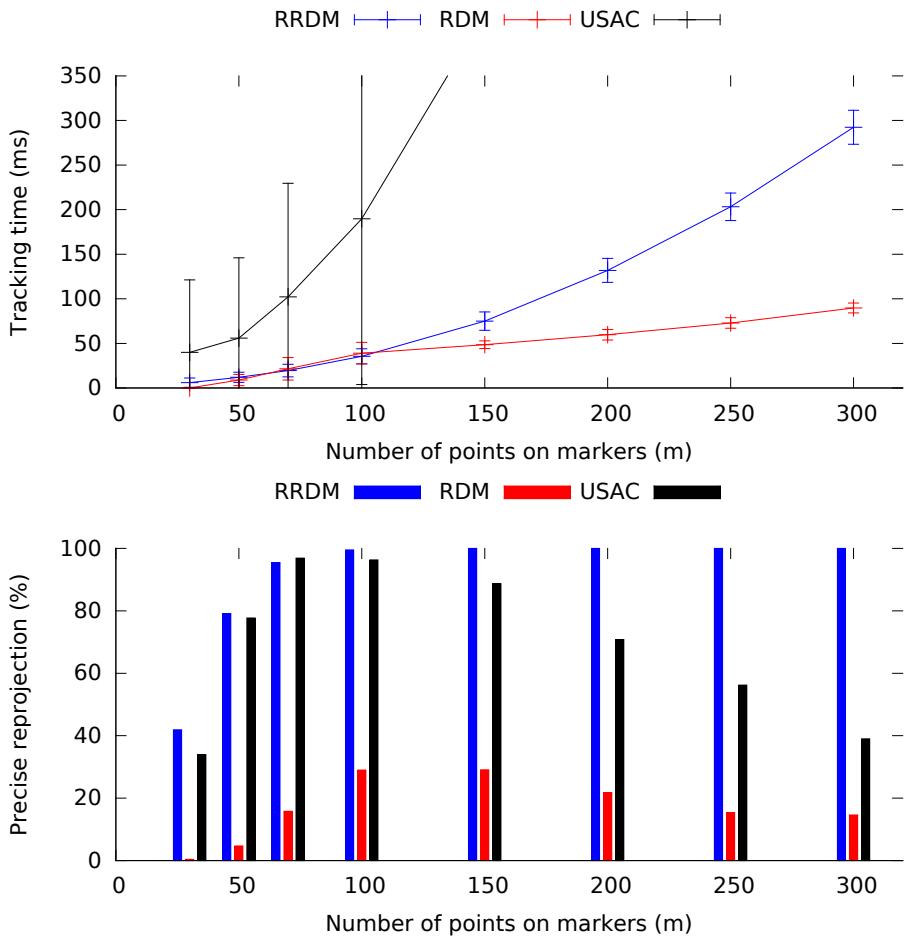


Figure 3.9: Point quantity experiment: robustness and speed with respect to m. $\beta = 15\%, \eta = 3\%$.

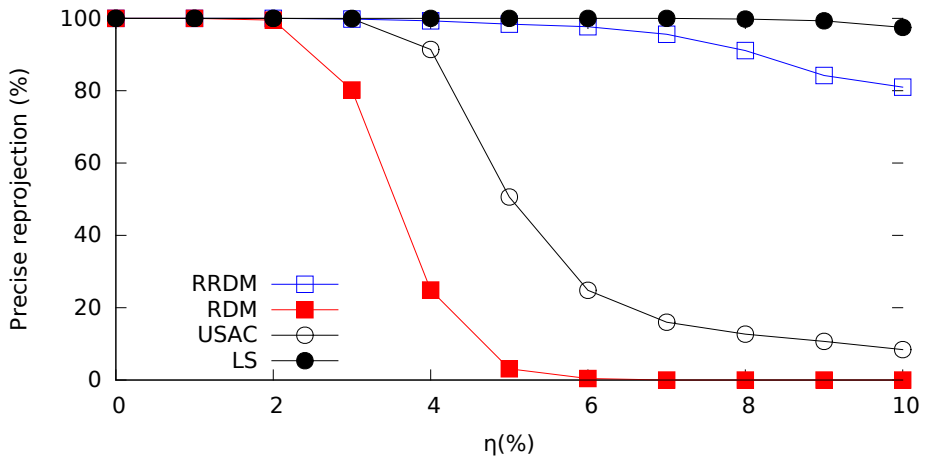


Figure 3.10: Jitter experiment: $m = 100$.

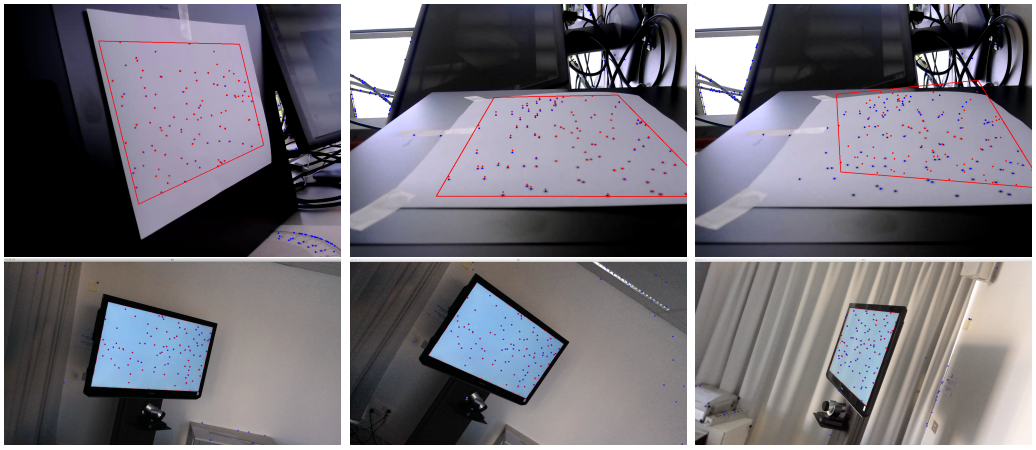


Figure 3.11: Video snapshots: upper band for paper markers (the rightest one did not work), lower band for screen-based markers.

simulate noisy points. Results are shown in Fig. 3.12. RDM's performance decreases significantly with respect to our proposal. USAC gives many results when β is high but most of which are wrong estimations.

In the "Tilted view experiment" (cf. Fig. 3.13), we compare the efficiency of the three methods on the basis of the perspective angle ψ, which is defined as the angle between the marker plane and the camera focal plane. As a consequence, a top view gives $\psi = 0°$. RRDM and USAC give better estimations than RDM, especially under large perspective angle $\psi \geq 70°$. At $\psi = 75°$, USAC seems to outperform RRDM, but it has too many incorrect reprojections (represented by the empty bar) to be used.

The "Missing point experiment" is dedicated to incomplete markers detection and tracking. There can be two kinds of missing elements in markers: (i) a whole part of a marker is invisible (occluded by another real object for example) and (ii) randomly distributed missing points are removed from the marker. Fig. 3.14 shows the results of experiments where (left) a marker is gradually occluded by a sheet of paper sliding on the screen and (right) where random points are removed from the marker. In both cases, our algorithm behaves slightly better, although it is a bit more significant in the randomly missing points case. The performance decay is logical since all algorithms use neighborhood relationships, which are quickly altered by random removals of points.

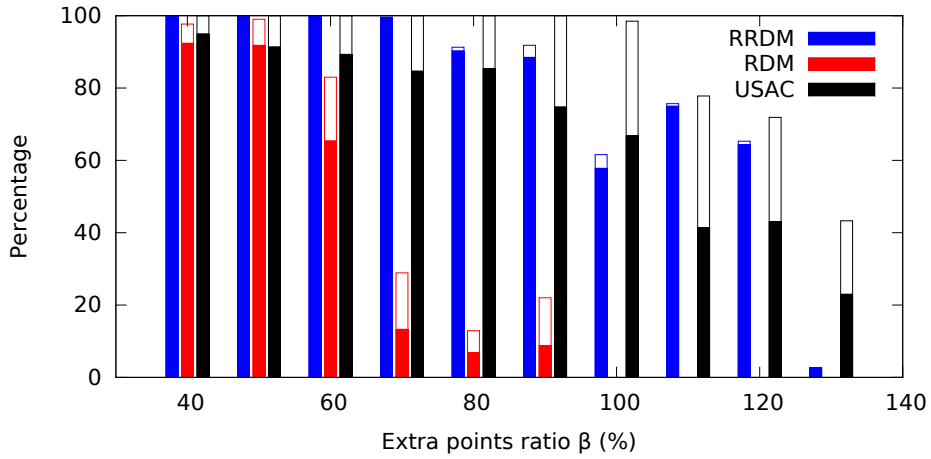


Figure 3.12: Extra point experiment, $\beta \times m$ extra points are added inside the marker. Solid bars represent correct reprojections, empty bars represent incorrect ones.

3.4.3 Natural map tracking

We apply RRDM to a map geo-referencing problem in the "Map tracking experiment". Three different unprepared printed maps of downtown Edinburgh containing different textures are used, cf. Fig. 3.15a-3.15c. We manually extracted from map (a) 233 intersections coordinates, which serve as the model point set (i.e. P). For each map, a video sequence containing about 100 frames is recorded with an iPad (with resolution 1280×720). Although only one geo-referencing operation is needed for each sequence of video in order to initialize tracking (if no tracking failure appears), we choose to apply it to each frame of the videos to show RRDM's behavior under diverse circumstances. For each frame, an automatic intersection detection technique based on color segmentation (similar to [Callier 2011]) with the help of manually generated filter offline is used to produce Q on the fly. In this experiment η is set to 5%.

To give an idea of the complexity of the problem we tackle, we manually analyzed the detection results on 10 randomly picked frames for each video (cf. Table 3.1). We consider a detected point to be a real intersection if it does not lay too far (closer than roughly 3 times the local road width) from any intersection on the map, otherwise it is considered to be a noisy point. However, a real intersection point in other maps may have no corresponding point in map (a) (i.e. P) as well. This can happen because different printed maps may not have the exact same road network. As a consequence, new road intersections may arise and the total number of inliers remains unknown. Detection for map (a) has a fairly good quality thanks

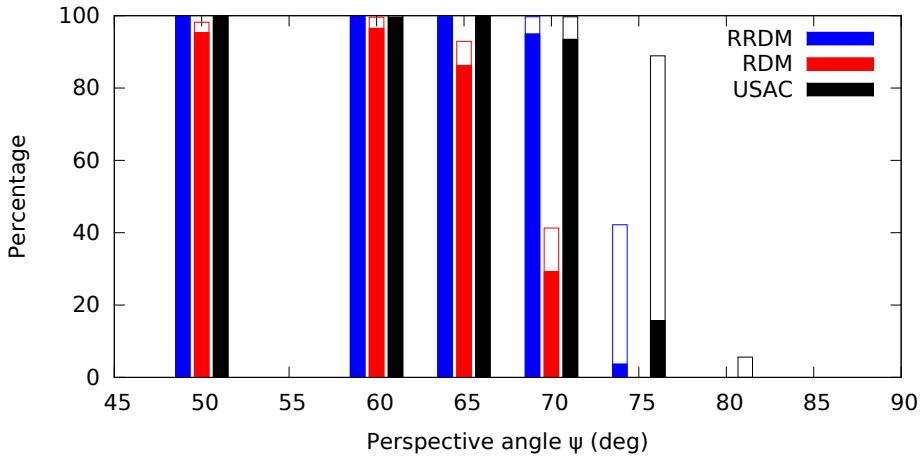


Figure 3.13: Tilted view experiment. Solid bars represent correct reprojections, empty bars represent incorrect ones.

Table 3.1: Qualities of detections in map tracking experiment (Mean value ± standard deviation).

Map ID	Point pattern size	Noisy points	Inliers found
(a)	236.7 ± 14.7	52.7 ± 5.0	102.0 ± 28.1
(b)	279.6 ± 16.9	114.9 ± 11.9	76.8 ± 19.8
(c)	237.1 ± 10.3	51.7 ± 7.4	56.9 ± 5.5

to its uniform road color (white) and high image quality (originally 1171×795 pixels). Roads in map (b) are white and yellow, those colors are similar to some background elements, leading to a much noisier detection result. Map (c) is also challenging since its original quality is low (553×461 pixels). Furthermore, scaling is not homogeneous between maps (c) and (a) (remember that P was constructed based on map (a)) thus preventing some inliers to be reprojected close enough to their corresponding detected points. This prevents the result list from a significant expansion during the recovery phase (cf. Section 3.2.2.3), as a consequence, correct reprojections are confined in a subregion of the map (see top left area in Fig.3.15c).

Tracking results of RRDM for the videos of maps (a), (b) and (c) are presented in Table 3.2. As both the total number of inliers and the ground truth of the homography are unknown, it is difficult to estimate whether a reprojection is correct. We consider a reprojection to be good if the basement of the 3D model of Edinburgh castle's 3D model is inside the castle area on the map which is mainly bounded by *Princes St., Kings Stables Rd, Johnston Terrace* and *The Mount* on the map. Note

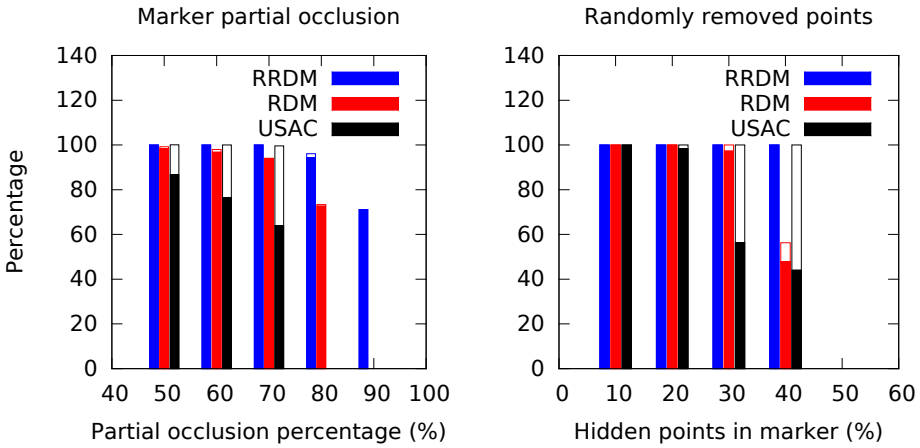
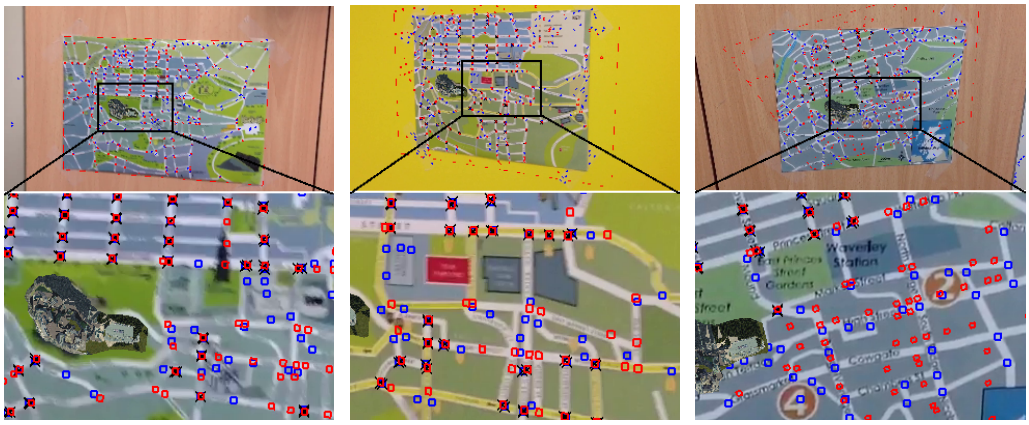


Figure 3.14: Missing point experiment. Solid bars represent correct reprojections, empty bars represent incorrect ones.



(a) (b) (c)

Figure 3.15: Augmenting natural maps of Edinburgh. Maps (a), (b) and (c) are three different maps registered with a single model and a 3D model of the castle is rendered onto them.

Table 3.2: Results of the "Map tracking experiment". Good matching rates is shown. Detection time and matching time per frame are reported in the form of "Mean value ± Standard Deviation".

Map ID	Good matching (%)	Detection time (ms)	Matching time (ms)
(a)	100.0%	73.0 ± 11.5	167.0 ± 14.2
(b)	90.2%	210.6 ± 23.8	223.6 ± 18.7
(c)	96.0%	81.2 ± 6.1	182.4 ± 15.4

that neither RDM nor USAC work for any of the videos.

3.5 Application: Augmented Maps

In this section, we propose a framework for augmented maps by using RRDM. It also includes a new method to better extract road network from images of paper maps without using manually generated filters. The work has been presented at the conference on *Machine Vision Applications (MVA)* in 2015.

The framework has three modules: road intersection detection, map-GIS registration and tracking. Reference intersections coordinates are calculated from GIS and cleaned beforehand since there are small roads which will never be present on a city map. During the initialization, three tasks are accomplished on the very first frame I_0: (1) Map intersections are detected from real maps by our road intersection detection module (cf. Section 3.5.1); (2) map intersections and reference intersections are matched by the map-GIS registration module using RRDM; (3) SURF key points from the paper map in I_0 are registered in the tracking module for later use. During online tracking, the paper map is tracked with a classical method based on SURF interest points and descriptors.

3.5.1 Intersection detection on real maps

Since the map-GIS registration is based on road intersections, the quality of road intersection extraction is crucial. False road intersections can lead to a poor performance of the method, or even to total failure. A usual way to find road intersections on a map is to identify the road network, extract its skeleton before finding lines intersections. The difficulty lies in the treatment of texts on maps, such as road names, which severely reduce the quality of the automatically extracted skeleton. [Callier 2012] relied on lines with homogeneous color in the image to find road pixels

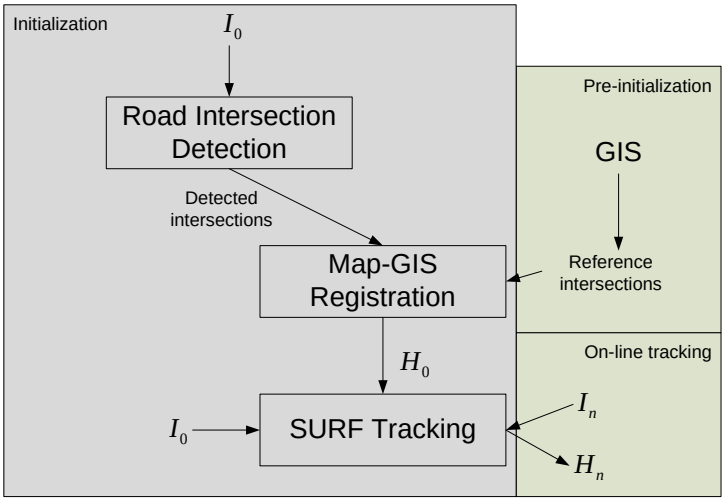


Figure 3.16: Schema of the application. I_0 is the first image used for initialization. H_0 is the homography between the GIS and the map contained in I_0

and used color histograms to find the road layer. [Chiang 2011] used Mean-shift, Median-cut and K-means to find the road layer and text/graph separation techniques are used. But none of these methods is efficient enough to allow for real-time augmentation. A modified version of the above methods is used in Section 3.4.3 but a map-dependent filter has to be used to speed-up the process.

The contribution of our new method is mainly about road layer extraction improvement, and is shown in Fig. 3.19. At first, the user picks a single road pixel from the image (a seed point). This is very easy to do with a mouse or on a tactile screen. From this pixel, road color and width can be estimated. Then, a classic level set method is used to find the map in I_0 to avoid false detections coming from out of the region of interest. At last, a modified level set method is applied to find the road layer inside the map.

We use a 50×50 local image portion centered at the seed point to find the road width since road width on a camera image of a map is usually far less than 50 pixels. A Canny edge detector [Canny 1986] is used to extract road edges and a flood fill at the seed point is applied to find the local road layer, which contains R road pixels. We then dilate the local road layer by 1 pixel and the result contains Q road pixels. Since roads can be modeled as thin rectangles, road width w can be easily calculated as:

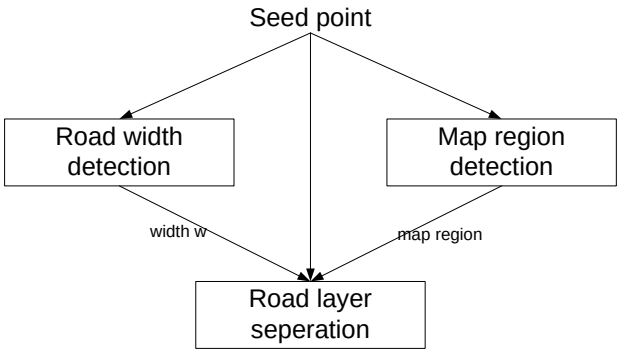


Figure 3.17: Workflow of road layer extraction

$$w = \frac{w_1 + w_2}{2} \quad \text{with } w_1 = 2R/(Q-R)\ ,\ w_2 = 2R/(Q-R+4) \tag{3.20}$$

where w_1 is for the case where a single road is present in the local image portion while w_2 is for the case where a crossing is present. These are the two most common cases in a local map.

The level set method, commonly known as active contour, is often used in image segmentation. It separates an image into different segments (i.e. subregions) such that colors of all pixels in the same segment are similar. It usually starts with an initial contour Γ_0 and then moves it iteratively in order to minimize $E(\Gamma)$ in (3.21) where Γ is the contour which separates segments A_1 and A_2. Shi's implementation is used as our classic level set method [Shi 2008]:

$$E(\Gamma) = \int_{A_1} |I_0(x,y) - \mathbf{C_1}| + \int_{A_2} |I_0(x,y) - \mathbf{C_2}| \tag{3.21}$$

where $I_0(x,y)$ is the color at (x,y). $\mathbf{C_1}$ and $\mathbf{C_2}$ are expected colors for the two segmentations respectively.

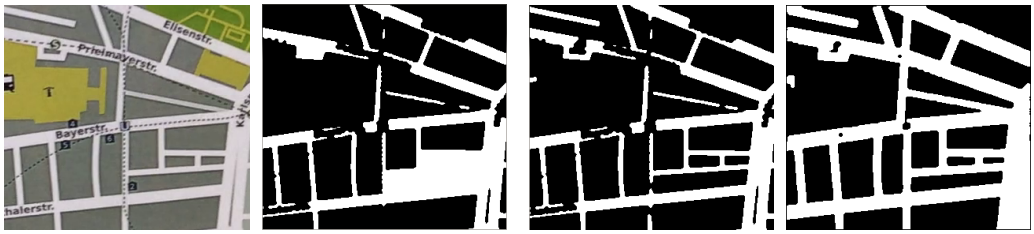
At first, a classic level set method is used to extract the map from the image I_0. An initial rectangle contour Γ_0 which is just a little smaller than I_0 is used (cf. Fig. 3.18a). C_1 and C_2 are the averages of pixels' values in segments A_1 and A_2 respectively. They change dynamically as the contour moves. Fig. 3.18c shows the segmentation result, with A_1 in white represents the map segment and A_2 in black represents the background .

Once we have got the map segment in I_0, a modified level set method is applied on this segment since the classic level set does not work well for road extraction.

(a) (b) (c)

Figure 3.18: Classic level set method for map extraction. (a) The initial contour Γ_0. (b) The final contour. (c) The result of two segments represented in different uniform colors A_1 (white) and A_2 (black).

More specifically, it has two major drawbacks: (1) It cannot segment holes from a region since the contour cannot appear from nowhere. However, road networks divide a map into many disconnected areas, thus forming "holes" (cf. Fig. 3.19b). (2) The resulting contours usually lie on pixels with high color gradients. So texts on the map can often prevent the contour from moving, which produces poor segmentation results (cf. Fig. 3.19c).



(a) (b) (c) (d)

Figure 3.19: (b)-(d) Results of various level sets. The segment A_1 represents road layer in white, the segment A_2 represents the background in black. (a) Original map portion. (b) Classic level set method. (c) The presence of road labels disturbs contours. (d) Our modified level set method.

To overcome the first problem, we initialize the level set method with rectangular stripes (cf. Fig. 3.20). Each stripe has a width of $2w$ cf. (3.20) and the initial contours are created from the edges of these stripes. This ensures that each "hole" is divided by the initial contours. For the second problem, since thickness of road labels is usually small, we can remove them by replacing each pixel by the pixel whose color is the most similar to road color (i.e. C_1) in the 5×5 neighborhood, cf. (3.22). Moreover, C_1 in (3.21)-(3.22) is set to the color of the seed point. C_2 is set to the average value of pixels in A_2.

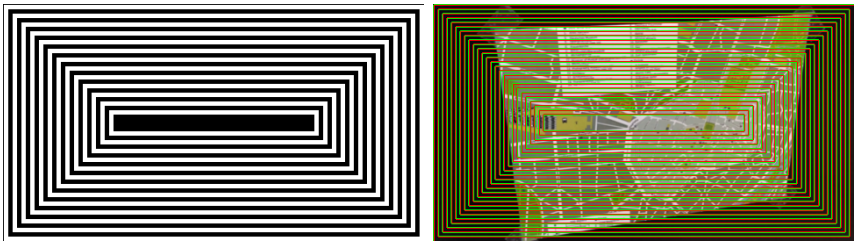


Figure 3.20: Left: an example of rectangular stripes. Right: Initial contours are created from the edges between white stripes and black stripes.

$$I_0(x,y) = \underset{|x'-x|\leq 2, |y'-y|\leq 2}{\arg\min} |I_0(x',y') - C_1| \tag{3.22}$$

3.5.2 Results

We recorded three videos of different paper maps of Munich with an iPad-air camera (with resolution 1920×1080) to test our method. We use OpenStreetMap data as our GIS database and raster maps found on the Internet and printed (see Fig. 3.21).

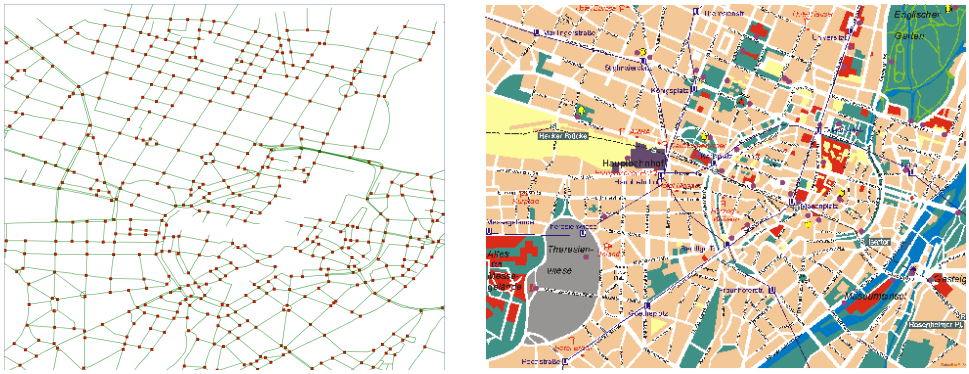


Figure 3.21: Data sources. Left: GIS road network (green) and road intersections (red). Right: One raster map.

We first compare results of intersection detection by the filter-based method applied in Section 3.4.3 and our modified level set method in Fig. 3.22. Detection results are greatly improved by our method while the filter-based method has many false detections especially near texts.

Initialization results for two videos are presented in Fig. 3.23. GIS roads are placed exactly onto roads of paper maps. Several tracking frames of the video are presented in Fig. 3.24. We can see that the position of the augmented information remains correct most of the time. But small drifts appear due to SURF tracking at

Figure 3.22: Intersection detection results (blue points). Top: filter-based method used in Section 3.4.3. Bottom: Modified level set (new method).

the end (cf. Frame 167 in Fig. 3.24). As to time performance, although registration takes 500∼1000ms, tracking works at interactive rate, i.e. 20∼30Hz.

3.6 Conclusion

In this chapter, we have introduced RRDM, a method to robustly track random dot patterns, where over and under point detection, large perspective distortion and inaccurate detection (jitter) can be taken into account. Unlike traditional feature point methods, RRDM is also able to robustly track 2D texture-less object. Recommendations on parameters are also given in order to facilitate user's implementation.

Synthetic images are used to demonstrate the robustness of RRDM against point jitter and when the marker is composed of a large number of points. In those cases, RRDM clearly outperforms RDM. We also proved through experiments with real videos that RRDM is more robust than RDM in presence of partial occlusion, randomly missing pattern points or under large perspective distortion, and especially with extra points in the pattern. Regarding efficiency, a real-time tracking (30Hz) can be achieved when the number of points in a marker is moderate (about 100 points). We also applied RRDM on registration of unprepared printed maps by using purely geometric coordinates of road intersections as the point pattern. Finally, we

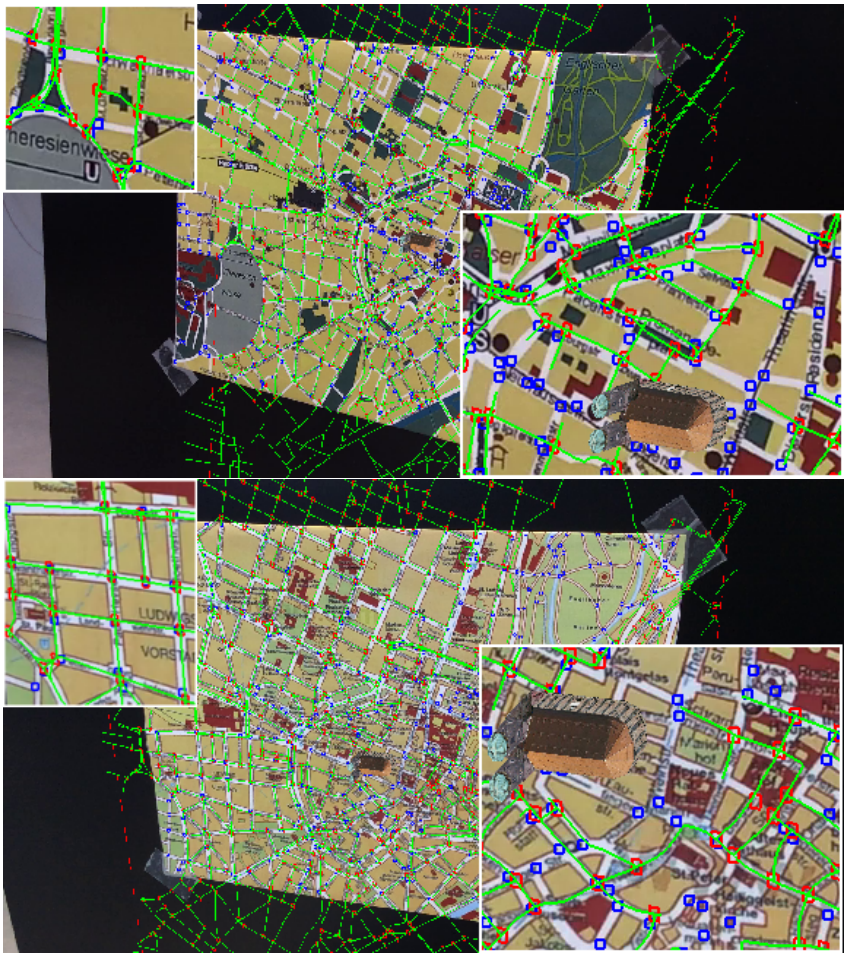


Figure 3.23: Initialization results for two different maps.

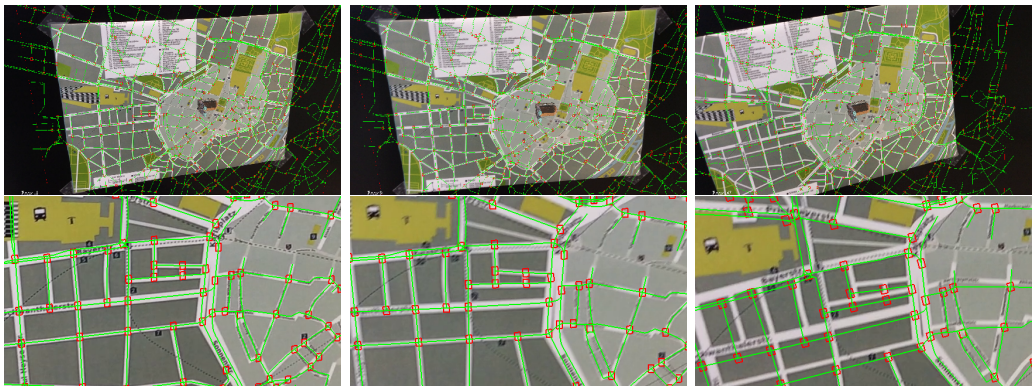


Figure 3.24: Frame 11, 94 and 167 (from left to right) of tracking results in video 3. Top row presents original augmented frames while bottom row shows zooms to illustrate the precision of the matching.

proposed a workflow for augmenting unprepared paper maps with a GIS database.

Although RRDM is robust and has almost solved one of the problems mentioned at the beginning of this dissertation (cf. Section 1.2), i.e. augmenting different paper maps without any preparation using GIS, it has three major drawbacks: (1) It can register the scene point set Q against only one model point set P but it cannot judge the correctness of a registration. Therefore, recognition of different point patterns cannot be performed. (2) It has quadratic time complexity in relation to the number of points to be tracked, which makes the algorithm slow when pattern contains more than 100 points. (3) It can only deal with 2D perspective transformations due to the TSR descriptor. In next chapter, we propose another algorithm which can overcome all these drawbacks.

Chapter 4

Local geometric consensus (LGC)

Contents

In this chapter, we consider the PPM problem between several model point sets and one scene point set. We present a method which can quickly and robustly match 2D and 3D point patterns based on their sole spatial distribution, but it can also handle other cues if available. This method can be easily adapted to

many transformations such as similarity transformations in 2D/3D, and affine and perspective transformations in 2D. This work has been presented at *International Symposium on Mixed and Augmented Reality (ISMAR)* in 2015.

We reformulate the matching problem as follows: consider having M model point sets (P_1, P_2, $\cdots$, P_M), each point set P_i containing m_i **uniformly randomly distributed points** in $d = (2, 3)$ dimensions. Let $T : \mathbf{R}^d \rightarrow \mathbf{R}^d$ be a geometric transformation on d dimensions of which we only know its type (similarity, homography, etc.). Let Q be a set of observed scene points, some of which belong to $T(P_i')$ where P_i' is a subset of P_i ($P_i' \subseteq P_i$) in which each point may have been slightly transformed (i.e. jitter). Other points of Q are extra points, i.e. acquisition noise. Our objective is to find P_i amongst the M known model point sets and to determine the transformation T between P_i and Q.

4.1 General algorithm

Our algorithm takes as inputs several model point sets P_1, P_2, $\cdots$, P_M, a query scene point set Q and the type of transformation. The output is the matched model number i ($i \in 1..M$) and T between P_i and Q, with T belonging to the given transformation type (cf. Fig. 4.1).

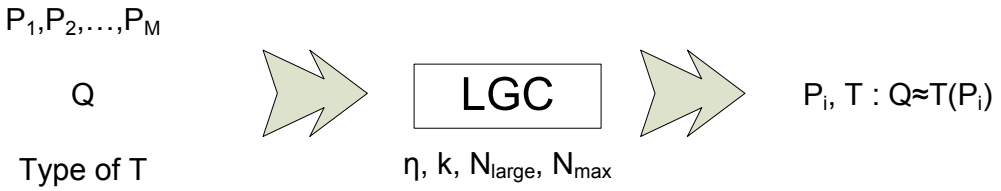


Figure 4.1: Input and outputs of our algorithm. η, k, N_{large} and N_{max} are the parameters of the algorithm.

The core idea is based on the consensus of local geometry: correspondences information between two small subsets of P_i and Q are used to estimate an initial guess of the transformation. The transformation is then used to find correspondences of other subsets in the neighborhood. At last the other correspondences between the two sets can be found iteratively. The very first correspondences between two subsets are found by Geometric Hashing (GH). More precisely, GH relies on local coordinate systems, which are invariant to the pre-defined transformation type, for matching.

Some important definitions and a brief description of LGC are given in Sec-

tion 4.1.1 while the next sections are dedicated to a detailed explanation.

4.1.1 Definitions and a brief description of the algorithm

The algorithm works with local geometric features. By assuming that points are randomly distributed, the relative positions between points in a local point set is very characteristic of this point set as mentioned in previous chapters. For each model point $p \in P_i$ or scene point $q \in Q$, p (or q) and its k nearest neighbors (k being a parameter of the algorithm) are used as a local geometric feature, that we call a p-patch (resp. q-patch). We define two patches (p-patch, q-patch) with known point correspondences as a *paired-patch* (p, q) (cf. Fig 4.2). When point correspondences are known, one can find the transformation between the paired-patch. As transformations (such as homographies) may not be linear, they can be approximated by other linear ones (such as affinities) in a small local area by using their Taylor expansions (cf. Section 4.5.2). A local transformation T_L is used in local patches as a linear approximation of T.

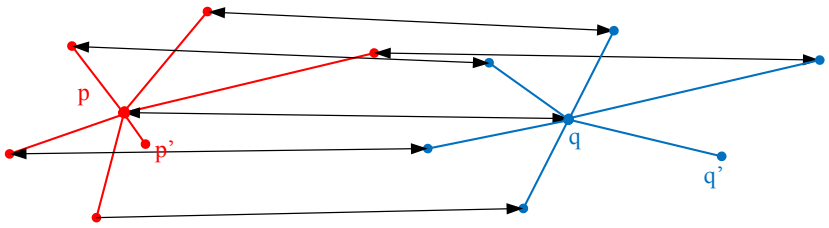


Figure 4.2: An example of a paired-patch (p, q) with $k = 6$. p and its 6 nearest neighbors are in red. q and its 6 nearest neighbors are in blue. Black lines with arrows represent correspondences of the paired-patch. Note that not all points have a correspondence due to missing/extra points, and wrong correspondences could also occur.

Let m_i be the number of points in model P_i and V_i be the space enclosed by P_i's convex hull. Then $\rho_i = \sqrt[d]{m_i/V_i}$ is the point density of P_i, where dimension d is the number of coordinates of each point. As a consequence, $l_i = \rho_i^{-1}$ is a measure of inter-point distance. Without losing generality, we assume that all point sets are normalized beforehand so that the center of mass of each point set is at origin and $l_i = l$ is now the same for all models. We assume also that the jitter follows an uncorrelated multivariate normal distribution $N(\mathbf{0}, \sigma\mathbf{I})$ for all models, where $\sigma = l\eta$ and η is the *jitter factor* (cf. Section 3.1.3). From now on, all point sets are normalized ones unless otherwise stated.

Our algorithm is composed of three modules: hypotheses generator, hypotheses validator, and results refiner (cf. Fig. 4.4). M containers C_i ($i = 1..M$) are used to store correspondences coming from each model P_i respectively.

The hypotheses generator is a two-stage procedure: offline registration (cf. Fig. 4.3) and online generation. During the offline stage, every model patch is registered in the generator. During the online stage, a scene point q is randomly chosen and the q-patch is fed to the hypotheses generator to build several paired-patches. These generated paired-patches are called "hypotheses" in the following. The validator takes each hypothesis as input and verifies them. If a hypothesis is judged as containing correct correspondences, the validator produces a list of correspondences and adds them to one of the containers C_i. The "generation-validation" process is looped until either one of the two following conditions is satisfied: (1) one container has more than N_{large} correspondences or (2) N_{max} scene points have been sent to the generator. We explain these conditions and their values in Section 4.1.5. Finally, the result refiner finds and improves the final result with the largest correspondences set among all containors $C_1..C_M$ (cf. Fig. 4.4). Note that only the hypotheses generator needs to work on all models whereas the hypotheses validator and the result refiner only use the scene point set Q and the model P_i which contains the model points of the input hypothesis.

The time complexity of the method is about $O(m + n)$, where $m = \sum_{i=1}^{M} m_i$ is the number of total model points, and n is the number of scene points. We use the term "about" since it is not formally proved but is only an estimation and validated empirically. It will be explained separately for each module in Section 4.1.2-4.1.4.

4.1.2 Hypotheses generator

The hypotheses generator is basically a geometric hashing module. During offline registration, each p-patch (i.e. a set of $k + 1$ points) is registered into a hash table along with its model number. During online generation, the input q-patch is considered as a query set and the corresponding p-patches that may belong to different models are retrieved.

We chose to use geometric hashing because it can drastically reduce the search space, it works well with moderate point jitter and in the presence of extra/missing points, and it can deal with many different types of transformations. By restricting it to local patches, the number of points in each p-patch is small enough for efficient matchings.

Geometric hashing uses a local basis. Let $f(T_L)$ be the degree of freedom of the

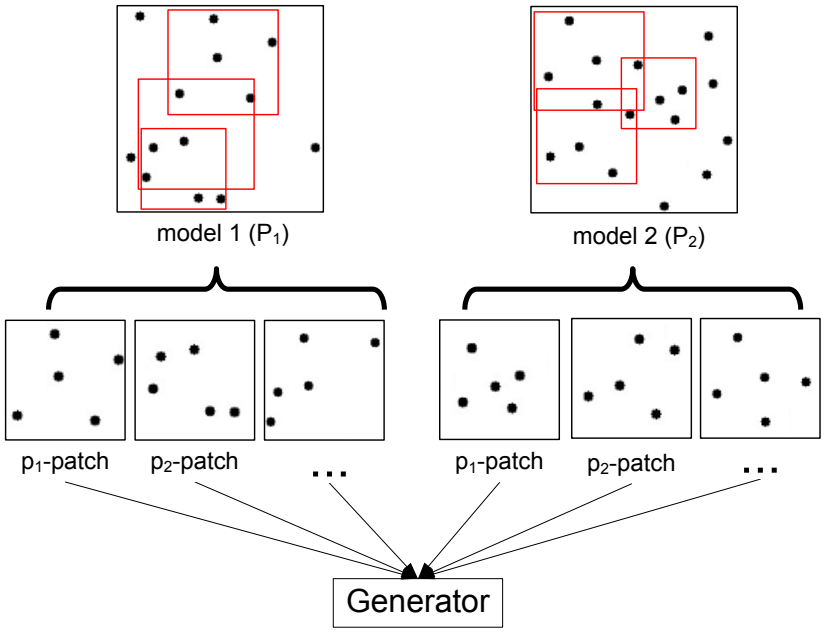


Figure 4.3: General schema for offline registration: p-patches are created for each point in all models and are registered into the generator.

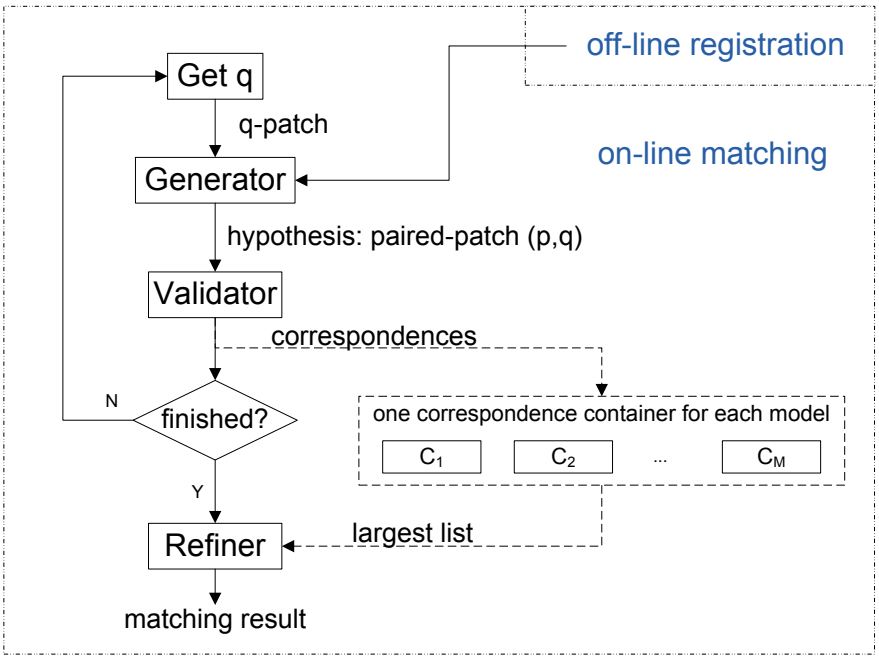


Figure 4.4: General schema for online matching: Generator (Section 4.1.2) is a geometric hashing module. Validator (Section 4.1.3) validates the input hypothesis and creates a list of correspondences from it. Refiner (Section 4.1.4) computes the final result.

linear transformation T_L. By using $b = floor(f(T_L)/d) + 1$ non-degenerated points, one can construct a local affine right-hand basis B as follows: let the first point be the origin of the local basis, each following point defines the direction of one axis for the local basis. Then the last point's position can be uniquely expressed by a geometric descriptor $\mathbf{X}$, which is a $D = bd - f(T_L)$ dimensional vector. The local basis is constructed so that $\mathbf{X}$ is invariant to transformation T_L. If $\mathbf{X}_{B_i}$ represents the descriptor in basis B_i formed by b points, we have:

$$\mathbf{X}_{B_1} = \mathbf{X}_{B_2} \Leftrightarrow \exists! T_L \text{ s.t. } B_2 = T_L(B_1) \tag{4.1}$$

where $\exists!$ represents the unique existential quantification operator meaning "there exists one and only one".

Jitter can influence the value of $\mathbf{X}$ as well. If the geometric descriptor of basis B is $\mathbf{X}$, and if B' which gives rise to a geometric descriptor $\mathbf{X}'$ is a jittered version of B, then $\mathbf{X}'$ is a random variable. Since T_L is a linear transformation, we can analytically find $\mathbf{\Sigma} = (\Sigma_1, \Sigma_2, ..., \Sigma_D)$ so that $X'_i \sim N(X_i, \Sigma_i^2)$ when the variance of jitter σ is comparably small to inter-point distance l. Note that for simplicity, $\mathbf{\Sigma}$ is not a covariance matrix but a vector of length D. As a consequence we say that B and B' with descriptors $\mathbf{X}$ and $\mathbf{X}'$ respectively, are matched under jittered conditions if (4.2) is satisfied:

$$|X_i - X'_i| \leq 2\Sigma_i \tag{4.2}$$

Given b points, there are many ways to construct a basis. For a p-patch, we choose p as the first point (i.e. the origin), other points are chosen so that $\mathbf{X}$ lies in a bounded region $R_X = \{\mathbf{X} : |X_i| \leq 1, 1 \leq i \leq D\}$. This region is uniformly partitioned into 100^D bins (meaning that [-1,1] in each dimension is equally divided in 100 segments) to serve as hash entries. To avoid inefficient record in the hash table, $\tilde{\Sigma}_i$ is used instead of Σ_i in Alg. 5, with $\tilde{\Sigma}_i = max(\Sigma_i, 0.05)$.

Algorithm 5 Offline registration for all p-patches in model P_i

for $B \in$ {All combination of b points in p-patch containing p} **do**
  Compute geometric descriptor $\mathbf{X}$ and its variance $\mathbf{\Sigma}$
  Register (model number i, B) into all bins that intersect $[\mathbf{X} - 2\tilde{\mathbf{\Sigma}}, \mathbf{X} + 2\tilde{\mathbf{\Sigma}}]$

Given the randomly selected input scene point q, the generation step begins by creating a list of (p_j-patch, q-patch) with all model points $p_j \in \cup P_i$ ($i = 1..M$). They are called pre-hypotheses since points correspondences are unknown in these two patches. Each pre-hypothesis has a local voting table for its neighbor points

correspondences. Then the voting schema detailed in Alg. 6 is applied.

Let us take one pre-hypothesis (p-patch, q-patch) as an example to show the creation of an hypothesis from the voting results. The process is similar to Alg. 3. First, all votes less than v_t (set to 4) are set to 0. Then point correspondences $s_{p,q}$ are estimated by solving the maximum assignment problem. To reduce the presence of bad hypotheses, the local transformation T_L is only estimated when the number of correspondences $size(s_{p,q})$ is larger than b, i.e. the number of points in basis B. $s_{p,q}$ and T_L are considered valid if they satisfy (4.3), where $f(T_L)$ is the degree of freedom of T_L. If so, $s_{p,q}$ is used to create the *paired-patch*(p,q), thus giving rise to one hypothesis.

$$\sum_{(\tilde{q},\tilde{p})\in s_{p,q}} ||T_L^{-1}\tilde{q}-\tilde{p}||^2 < \chi^2_{0.05,size(s_{p,q})-f(T_L)} \tag{4.3}$$

Algorithm 6 Online hypotheses generation with q-patch

Input: A randomly selected scene point $q \in Q$
Output: A list of hypotheses: *hypo*
Create (p-patch, q-patch) pre-hypotheses, $p \in \cup P_i, i = 1..M$
for all basis $B^* \in \{q\text{-patch}\}$ **do**
  Find geometric descriptor **X**
  for all (model number i, B) in the hash bin which covers **X** **do**
    Let p = the origin point of B
    Cast one vote for each correspondence between B^* and B in the local voting table of (p-patch, q-patch)
for all pre-hypotheses (p-patch, q-patch) **do**
  Set votes less than v_t to 0
  $s_{p,q}$ = Solution of maximum assignment problem of the voting results (0 vote excluded)
  if $size(s_{p,q}) > b$ **then**
    Estimate T_L using $s_{p,q}$
    if $s_{p,q}$ and T_L satisfy (4.3) **then**
      hypo $\leftarrow$ paired-patch (p,q)

Since the generator is a multi-model GH module, its time complexity is $O(m(k+1)^{2b})$ in the worst case when all p-patches are hashed in a few bins [Wolfson 1997], where m is the total number of model points, k is the number of nearest neighbors used and b is the number of points in B, which only depends on the local transformation type T_L.

4.1.3 Hypotheses validator

The core work of the validator is to check whether the local transformation T_L of an input hypothesis is an approximation of T. If a paired-patch contains point correspondences of the real transformation, it is called an *in-paired-patch* and its local transformation is a good approximation of T. Otherwise, it is called an *out-paired-patch*.

Local transformations of two neighboring in-paired-patches should be similar (cf. Section 4.2 for details) as they are both approximations of T in the same neighborhood. Such a relationship cannot be established for out-paired-patches. We use this consensus between neighbor paired-patches' transformations to validate hypotheses. Obviously, the more paired-patches share a consensus with T_L, the more likely T_L is a good approximation of T. The measurement of consensus is T_L-dependent, it will be addressed when we discuss concrete implementations, cf. Section 4.2.

The idea of consensus between neighboring paired-patches was already seen in RRDM (Chapter 3), however this approach is different. In RRDM, transformations of neighboring paired-patches are first estimated independently by voting before being compared, which consumes a lot of time. Furthermore, when there are lots of noise, voting results will become unreliable and thus correct point correspondences can hardly be found only via voting. In LGC, if the transformation of one paired-patch is found by voting, we use it as an initial guess to estimate transformations of its neighboring paired-patches. This allows us to improve the possibility and the speed of finding similar transformations between neighboring paired-patches. In addition, by relying on the consensus and on a proper termination criterion N_{large}, LGC can almost guarantee to find a correct estimation of T (if it exists), which will be further improved by the refiner module.

Alg. 7 describes the validator module. Recall that a paired-patch contains correspondences between two patches, thus it can be seen as a small correspondence list as well. The validator takes an hypothesis paired-patch (p_0, q_0) as input, finds its neighboring paired-patches having a similar local transformation. Since these neighboring paired-patches are in geometric consensus with the hypothesis, they can be seen as supporters of it. They are added to a *supporterlist*. Then the validator module does the same thing to these newly added paired-patches (i.e. supporters). This process continues until no paired-patch can be added anymore. The resulting *supporterlist* is a list of correspondences in the same subregion and in geometric consensus. If the local transformation T_L of the input hypothesis is considered as a

good approximation of T, supporterlist is registered into container C_i, i being the model number of point p_0. If C_i is not empty, supporterlist and the stored list are merged to give a larger correspondences list sharing the same T.

Algorithm 7 Validator

Input: One hypothesis: paired-patch (p_0, q_0)
Output: A list of correspondences: *supporterlist*
supporterlist = $\{(p_0, q_0)\}$
for each $(p, q) \in$ *supporterlist* **do**
  T_L = local transformation of paired-patch (p, q)
  for all $(p', q') \in$ paired-patch (p, q) **do**
    Find correspondences $s_{p',q'}$ of (p'-patch, q'-patch) by using nearest neighbors between p'-patch and $T_L^{-1}(q'-\text{patch})$.
    if $size(s_{p',q'}) > b$ **then**
      Estimate T'_L using $s_{p',q'}$
      if $s_{p',q'}$ and T'_L satisfy (4.3) **and** $T_L \simeq T'_L$ **then**
        supporterlist $\leftarrow$ *supporterlist* $\cup\{(p', q')\}$

Time complexity of the validator depends on the number of input hypotheses. In the worst case, N_{max} scene points have been fed to the generator and each scene point generates m hypotheses, thus generates mN_{max} hypotheses. As a consequence, the time complexity is $O(mN_{max})$ in this worst case.

4.1.4 Result refiner

After loops of "generator-validator", many correspondences are filled into containers. If the container C_i has the most correspondences, then model P_i is considered to appear in the scene. The refiner takes the correspondence list in C_i as input, noted as sl. It works on the scene point set Q and the model point set P_i to find transformation T. The overall refining process is similar to Section 3.2.2.3, but *Delaunay triangulations* and a threshold d_t are used to improve its performance.

First of all, Delaunay triangulations are generated for both point sets. A point p's *mesh neighbors* are defined as the points who directly connect to p by Delaunay mesh (cf. Fig. 4.21). As a consequence, each point is connected with its mesh neighbors by the Delaunay mesh. From sl, a transformation T can be estimated. Then we try to find matching points for all mesh neighbors of points in sl. When a new matching is found, the new correspondence is added to sl. With these correspondences, the transformation can be better estimated. This process continues until sl stops growing. Note that at the end of Alg. 8, T is the transformation between normalized P_i and Q. However, since point correspondence between original

sets can be derived from sl, it is very easy to calculate the transformation between them. The reason for using mesh neighbors instead of nearest neighbors is explained in Section 4.5.1.

Algorithm 8 Refiner

Input: Largest correspondence list: sl
Output: Transformation: T
repeat
  Estimate T using sl
  for all $(p, q) \in sl$ **do**
    Add all mesh neighbors of q in *potentialComers*
    Add all mesh neighbors of p in *tmpSet*
    In Q, find nearest neighbors of each $T(p_j), p_j \in tmpSet$
    Add these nearest neighbors into *potentialComers*
    for all $q' \in potentialComers$ **do**
      if $\exists p' \in P_i : |p' - T^{-1}(q')| \leq d_t$ **then**
        $sl = sl \cup \{(p', q')\}$
  Estimate T using new sl
  Erase $(p, q) \in sl$ if $|p - T^{-1}(q)| > 3\sigma$
until sl stops growing
Return: T

In Alg. 8, d_t is a threshold which describes how far could a projected scene point $T^{-1}(q')$ be away from its model corresponding point. This value can vary according to the position of q'. Let C be the convex hull of scene points in sl. If q' is inside C, we can take roughly $d_t = 2\sigma$ with σ being the variance of the additive Gaussian noise. When q' is outside C, the farther it is from C, the larger d_t will be (cf. Fig. 4.5). Assuming o is the center of C and C intersects $\overline{oq'}$ at q_C, we then take $d_t = 2\frac{\overline{oq'}}{\overline{oq_C}}\sigma$.

In the worst case, each repeat loop only adds one correspondence into sl, which means that repeat loop runs at most n times (i.e. the number of points in Q). Thus the time complexity in the worst case is about $O(n)$. We use the term "about" since we consider the inner loop is fast enough and thus not influence much the refiner module.

4.1.5 Parameters

All parameters used in LGC will be discussed here so that they can be more easily adjusted according to specific use.

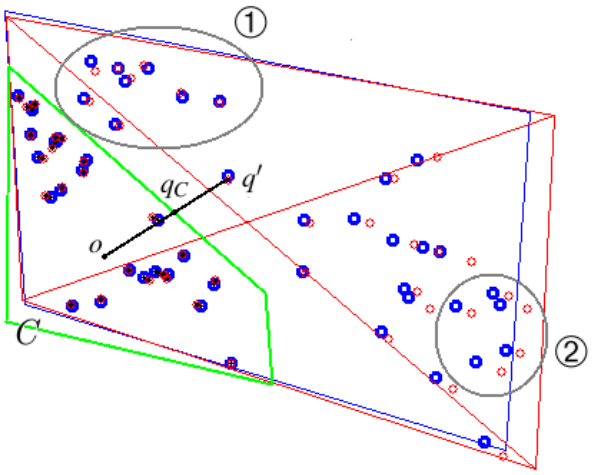


Figure 4.5: An illustration before the refiner module: model points (in red) are projected into the scene. Scene points are in blue, correspondences in *sl* are in black. The convex hull of scene points in the correspondence list C is represented by a green polygon, o being its center. Region ① is near C and thus the projected model point and the scene point almost overlap. Region ② is far from C and thus the distance between the projected model point and the scene point is larger. A correspondence will be established for point q' outside of C if a model point is projected within $d_t = 2\frac{\overline{oq'}}{\overline{oqC}}\sigma$

4.1.5.1 k and η

The number of nearest neighbors k defines the size of local patches, the bigger it is, the more robust the algorithm is against extra/missing points. Since our method is based on Geometric Hashing, the efficiency related to k in the generator is $O((k+1)^b)$, where b is the number of points in a basis B (cf. Sec. 4.1.2).

$\eta = \sigma/l$ is the jitter factor. When it is larger, the algorithm is more robust against acquisition noise, but each p-patch will be registered into more hash bins as well. The latter impacts the efficiency of the generator at $O(\eta^D)$, where D is the dimension of the geometric descriptor (cf. Sec. 4.1.2). We found that $\eta \approx 10\%$ is the limit for the jittered point pattern matching problem since beyond that value, precise transformation is difficult to obtain even with pre-known point correspondences (cf. Fig 4.10). Fortunately, we found $\eta = 5\%$ - 7% to be sufficient for many applications.

4.1.5.2 N_{large} termination rule

The first termination condition on N_{large} means that we are almost sure to have found the model so there is no need to test more hypotheses. A good choice should

give very few false alerts. Given a model P_i and the scene Q, let us consider two random events E_0 and E_1:

$$\begin{aligned} E_0 &= \quad Q \text{ is a random set. (i.e. } Q \text{ does not come from } P_i) \\ E_1 &= \quad \exists N_{large} \text{ consensus correspondences in a local region between } Q \text{ and } P_i \end{aligned}$$

Reducing false alerts means reducing the probability of E_0 knowing E_1, i.e. $P(E_0|E_1)$. We can show that $P(E_0|E_1) \approx 0$ with a very small N_{large}. In a conservative estimation (see Section 4.1.6 for demonstration) where one has 10^4 models to track and each model has 10^6 points, $P(E_0|E_1)$ is smaller than 10^{-6} with $N_{large} = 20$. However, this result cannot guarantee that the method can achieve an accurate estimation of T when a model is present, because the refiner may find false inliers (cf. Fig. 4.6).

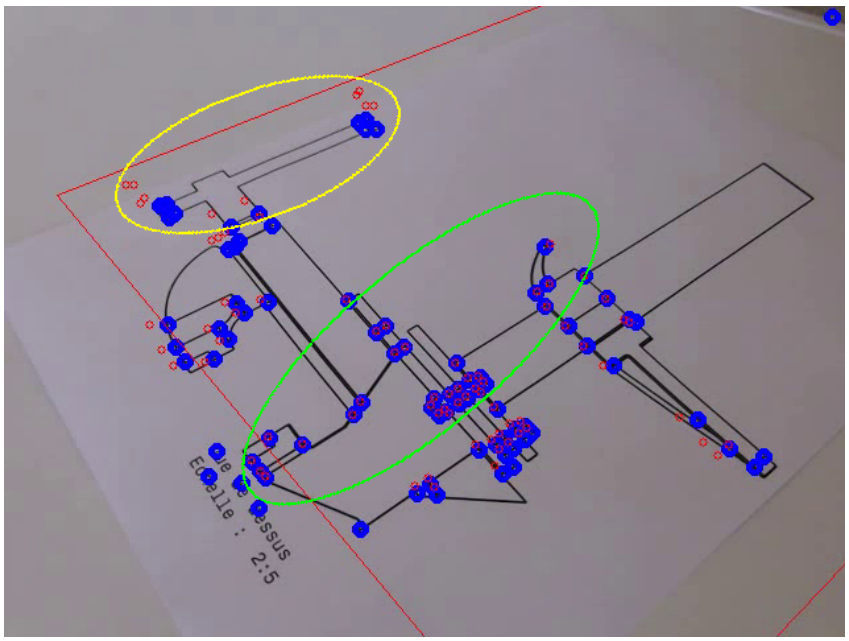


Figure 4.6: Condition N_{large} confirms the presence of model but refiner finds outliers so the estimation of T is not accurate. It can be seen in the yellow ellipse where points are correctly detected but not matched. Correspondences found in validator are inside the green ellipse and are inliers.

4.1.5.3 N_{max} termination rules

The second condition on N_{max} means that we are almost sure that no model exists for the point set Q. This is a direct inspiration from RANSAC. If Q has t inliers, and if an in-paired-patch has a probability p_1 to be proposed to the validator and a probability p_2 to pass the validation phase, then the probability (λ) of no result from the validator after trying N image points is:

$$\lambda = (1 - \frac{t}{n} p_1 p_2)^N \tag{4.4}$$

If we want the probability of no matching results λ be smaller than a predefined value λ_{max}, i.e. $\lambda < \lambda_{max}$. We have:

$$N_{max} = \frac{\log(\lambda_{max})}{\log(1 - \frac{t}{n} p_1 p_2)} \tag{4.5}$$

From our experiences, the correct correspondences in an in-paired-patch may be quite difficult to find in very noisy cases. So p_1 can be very small. However, once an in-paired-patch is found, it is easy to find other in-paired-patches in the neighborhood and thus it will probably pass the validation. As a consequence, p_2 is large. Let us take $p_1 = 0.2$ and $p_2 = 0.7$. In a scene with 50% inliers, if we want $\lambda < \lambda_{max} = 0.05$, we will get $N_{max} = 41$. So no matter how large the point set in the scene, one needs only to do at most 41 simple geometric hashing queries with $k+1$ points. In all our experiments, N_{max} is set to be 45. One can adjust N_{max} to a more proper value according to (4.5) to adapt to different outlier ratios.

4.1.6 Local consensus: a guarantee of low false alert

We want to find probability of false alert $P(E_0|E_1)$ mentioned in Section 4.1.5.2. According to Bayes' theory, we have:

$$P(E_0|E_1) = \frac{1}{1 + \frac{P(E_1|\overline{E_0})P(\overline{E_0})}{P(E_1|E_0)P(E_0)}} \tag{4.6}$$

To have a conservative estimation, we want to find the upper bound of $P(E_0|E_1)$. If the upper bound is small enough, it is less likely to have false alerts. $P(E_1|\overline{E_0})$ and $P(E_1|E_0)$ are two events independent of $P(\overline{E_0})$. So $P(E_0|E_1)$ will be larger if $P(\overline{E_0})$ is smaller. $P(\overline{E_0})$ can be seen as the probability that model P_i appears in the scene. When there are more models to track, $P(\overline{E_0})$ will be smaller. Considering we have 10^4 models to track, which is a huge number, the probability that P_i appears

in the scene is roughly 10^{-4}. In (4.6), $P(E_1|\overline{E_0})$ is very large, say $P(E_1|\overline{E_0}) \approx 1$. We have to find $P(E_1|E_0)$.

Before solving $P(E_1|E_0)$, let us consider a simpler problem (the problem of long runs): In a N_b random binary string, 1 appears at each bit with probability p_b. The length of the largest sequence containing only 1s is a random variable L. Then, what is the expectation and the variance of L? Schilling et al. [Schilling 2012] show that the result can be approximated by the following equations:

$$\begin{aligned} E(L) &= log_{1/p_b}\left(N_b(1-p_b)\right) \\ \Sigma(L) &= \frac{\pi}{\sqrt{6}ln(1/p_b)} \end{aligned} \tag{4.7}$$

where the expectation $E(L)$ is proportional to $logN_b$, it varies little with respect to the length of the sequence. The variance $\Sigma(L)$ is independent of N_b.

If we define random field as a field containing many independent and identically distributed (i.i.d) random experiments organized in space. The problem of long runs describes the probability that all experiments in a region of the random field have the same experiment results. $E_1|E_0$ describes the probability that the most of experiments (instead of all experiments) in a region of the random field have the same experiment results. Both problems have a similar nature, thus (4.7) can be used for a rough estimation.

For a given transformation T, a point correspondence is considered coherent with T if the scene point is back-projected within a 3σ radius neighborhood of the model point (cf. Alg. 8). If the scene point has a random position, the probability of the coherence is about $(3\sigma/l)^d$. Remember that l is the "inter-point" distance defined in Section 4.1.1. We choose the limit of jittered point pattern matching, $\sigma = 0.1l$ (cf. Section 4.1.5). If at least every two correspondences contain one inlier and $d = 2$, which is a conservative estimation, this results in $p_b \approx 0.2$. Let $N_b = 10^6$ points, we have $E(L) \approx 8.4$ and $\Sigma(L) \approx 0.8$.

If the transformation T needs b_T points as basis which gives us b_T "free" points in N_{large}, we have:

$$P(E_1|E_0) = P(L \geq N_{large} - b_T) \tag{4.8}$$

Take as example a 2D homography (i.e. $b_T = 4$) and $N_{large} = 20$, thus $N_{large} - b_T - E(L) \approx 7.6 \approx 9.5\Sigma(L)$. [Schilling 2012] shows that the distribution of L is approximately a normal distribution. As a consequence, $P(E_1|E_0) = P(L \geq E(L) + 9.5\Sigma(L)) \ll 10^{-10}$. Thus $P(E_0|E_1) \approx 10^{-6}$ and it is a conservative upper

bound estimation.

4.2 Specific Implementations

In this section, we present the detailed implementation of the algorithm for 2D homography and 3D similarity. 2D homography is used to match coplanar point sets (markers) while 3D similarity matching is very useful in 3D model registration. Given the general algorithm, one needs three more pieces of information for a detailed implementation: (a) How to construct the basis B and compute the geometric descriptor $\mathbf{X}$ (b) How to measure the similarity between T_Ls. (c) How to find T_L or T with known point correspondences. We detail each of them for both cases.

4.2.1 2D homography

A 2D homography is basically a perspective transformation, and each point has only two coordinates. So T belongs to 2D perspective transformation and T_L belongs to 2D affine transformations.

(a) The basis B is ordered as (p_0, p_1, p_2, p_3) so that $\Delta p_0 p_1 p_2$ has the largest surface among all triangles containing p_0. p_1 and p_2 are chosen so that $\overrightarrow{p_0p_1} \times \overrightarrow{p_0p_2} > 0$. This gives us a local right-hand basis and $|X_1|, |X_2| \leq 1$ (cf. Fig. 4.7), same as in [Lamdan 1990].

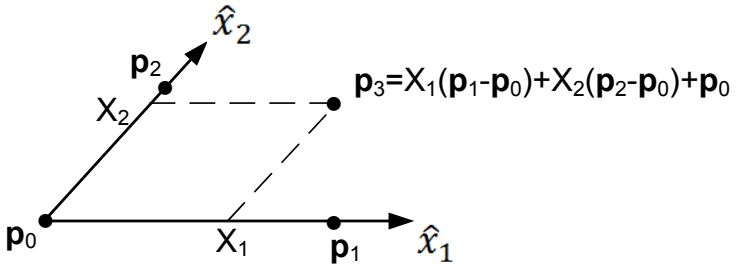


Figure 4.7: Local basis B (p_0, p_1, p_2, p_3) for 2D homography. p_0 is the origin. $\hat{x}_i$ are axes of the local coordinate system.

Thus, if $p^{(i)}$ stands for the i-th coordinate of point p, we have:

$$\begin{aligned}
\mathbf{A} = (a_{ij}) &= \begin{pmatrix} p_1^{(1)} - p_0^{(1)} & p_2^{(1)} - p_0^{(1)} \\ p_1^{(2)} - p_0^{(2)} & p_2^{(2)} - p_0^{(2)} \end{pmatrix}^{-1} \\
\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} &= \mathbf{A} \begin{pmatrix} p_3^{(1)} \\ p_3^{(2)} \end{pmatrix} \\
\begin{pmatrix} \Sigma_1 \\ \Sigma_2 \end{pmatrix} &= ((X_1 + X_2 - 1)^2 + X_1^2 + X_2^2 + 1)\sigma \begin{pmatrix} a_{11}^2 + a_{12}^2 \\ a_{21}^2 + a_{22}^2 \end{pmatrix}
\end{aligned} \tag{4.9}$$

(b) A 2D affinity is a 2×3 matrix. After a singular decomposition of the left 2×2 square matrix, a rotation θ and two scale factors $\alpha^{(1)}, \alpha^{(2)}$ can be retrieved. By using the result from Section 3.2.2.2, we say that two affine transformations are similar if the difference of their rotation is less than $10°$ and the ratio of each scaling is less than 1.3.

(c) With known point correspondences, T_L can be found by the least square method while T can be found in various ways including OpenCV's *findHomography* method.

4.2.2 3D similarity

A similarity is a linear transformation itself, so $T_L = T$.

(a) A basis B is ordered as (p_0, p_1, p_2) so that $\|\overrightarrow{p_0p_1}\| \geq |\overrightarrow{p_0p_2}\|$. $\overrightarrow{p_0p_2}$ defines the positive direction of the second axis of this local basis (cf. Fig. 4.8). Thus the descriptor $\mathbf{X}$ (i.e. the coordinates of p_2) can be expressed as:

$$\begin{aligned}
X_1 &= \frac{\overrightarrow{p_0p_2} \cdot \overrightarrow{p_0p_1}}{\|\overrightarrow{p_0p_1}\|^2} \\
X_2 &= \sqrt{(\|\overrightarrow{p_0p_2}\| / \|\overrightarrow{p_0p_1}\|)^2 - X_1^2} \\
X_3 &= 0
\end{aligned} \tag{4.10}$$

It is easy to verify that $|X_1|, |X_2| \leq 1$ and thus variances can be computed as follows:

$$\begin{aligned}
\Sigma_1 = \Sigma_2 &= \frac{\sqrt{\|\overrightarrow{p_0p_1}\|^2 + \|\overrightarrow{p_0p_2}\|^2 + \|\overrightarrow{p_1p_2}\|^2}}{\|\overrightarrow{p_0p_1}\|^2} \\
\Sigma_3 &= 0
\end{aligned} \tag{4.11}$$

(b) A 3D similarity can be decomposed into a rotation matrix R and a scale α factor. R can be represented as a quaternion q. We say two similarities are equal if the difference of rotation is less than $10°$ measured by Φ_3 defined in [Huynh 2009]

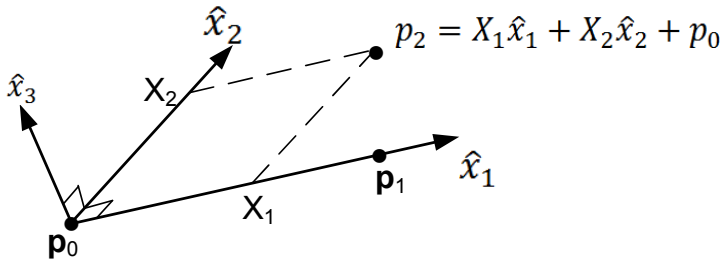


Figure 4.8: Local basis B (p_0, p_1, p_2) for 3D similarity. p_0 is the origin. $\hat{x}_i$ are axes of the local coordinate system, with $\hat{x}_i \perp \hat{x}_j$, $(i \neq j)$. X_i are the coordinates of p_2 under this local coordinate system, with $X_3 \equiv 0$.

and the ratio of scales is less than 1.3, that is:

$$2\arccos(\|q_1 \cdot q_2\|) \leq 10^\circ, max(|\frac{\alpha_1}{\alpha_2}|, |\frac{\alpha_2}{\alpha_1}|) \leq 1.3 \tag{4.12}$$

(c) We use the method from [Umeyama 1991] to estimate the similarity T with known point correspondences. Knowing n point correspondences (p_i, q_i), $p_i \in P$, $q_i \in Q$, $i = 1..n$, the rotation R, the scale s and the translation t minimize the sum of squared error $\sum_{i=1}^{n} \|q_i - (cRp_i - t)\|^2$ are:

$$\begin{aligned} R &= USV^T \\ s &= \frac{n}{\sum_{i=1}^{n} \|p_i - \bar{p}\|^2} tr(DS) \\ t &= \bar{q} - sR\bar{p} \end{aligned} \tag{4.13}$$

where U, V and D are the results of singular decomposition of covariance matrix Σ $(\Sigma = UDV^T)$, and

$$\begin{aligned} \bar{p} &= \frac{1}{n}\sum_{i=1}^{n} p_i \\ \bar{q} &= \frac{1}{n}\sum_{i=1}^{n} q_i \\ \Sigma &= \frac{1}{n}\sum_{i=1}(q_i - \bar{q})(p_i - \bar{p})^T \end{aligned} \tag{4.14}$$

$$S = \begin{cases} diag(1, 1, ..., 1, 1), & \text{if } det(\Sigma) \geq 0 \quad (4.15) \\ diag(1, 1, ..., 1, -1), & \text{if } det(\Sigma) < 0 \quad (4.16) \end{cases}$$

4.3 Results on synthetic point sets

In this section, we present results obtained under different conditions. Section 4.3.1 focuses on performance in noisy conditions. While this first section focuses on 2D homography and performs extensive tests, Sections 4.3.2 and 4.3.3 show that the algorithm has the potential to work well in 3D and can benefit from additional information. All experiments are performed on a PC with an Intel Xeon E3 1240@3.40GHz CPU and 16GB of RAM.

4.3.1 Speed and robustness studies

The performance of our method is compared with other methods using synthetic data. In the graphs, RDM stands for Uchiyama's Random Dot Markers [Uchiyama 2011b] (the original "tracking by detection" implementation is used), RRDM for the Chapter 3, LGC for our current proposal and GH for traditional Geometric Hashing [Wolfson 1997]. The default parameters of these methods are used. The 2D homography is set to a 30° perspective transform unless stated otherwise. Fig. 4.9 to Fig. 4.14 present single model matching, i.e. the scene set Q is matched against only one model set P, while Fig. 4.15 shows multi-model matching. We use (3.19) from Section 3.4.1 to judge whether reprojections are precise.

We first investigate the performance with different number of points in the model point set P. Fig. 4.9 shows the result in an ideal condition, that is without jitter ($\eta = 0\%$), extra or missing point, as well as the result with $\beta = 15\%$ extra points and with jitter $\eta = 3\%$. The second case is more realistic and can be seen as a simulation of a real scene. As shown in the figures, GH is too slow to provide results for more than 60 points. RRDM has a clear quadratic behavior. RDM works well in the ideal condition but finds too few precise results in the second case. LGC achieves the best performance with a linear behavior. We can notice that, LGC performs better than RRDM for few points in terms of reprojection precision. This may be counterintuitive since RRDM uses exhaustively local matching but gives worse results. In fact, this behavior can be understood if we look into details of both algorithms. RRDM calculates local transformations of subsets independently, so two local transformations are in consensus only if both of them are estimated correctly independently. But the validator of LGC calculates the local transformation of a subset by using the local transformation of its neighboring subset as an initial guess. Consensus between local transformations can be found once the first local transformation is estimated correctly. Therefore, LGC has more chance than RRDM to find subsets in consensus which give rise to more inliers.

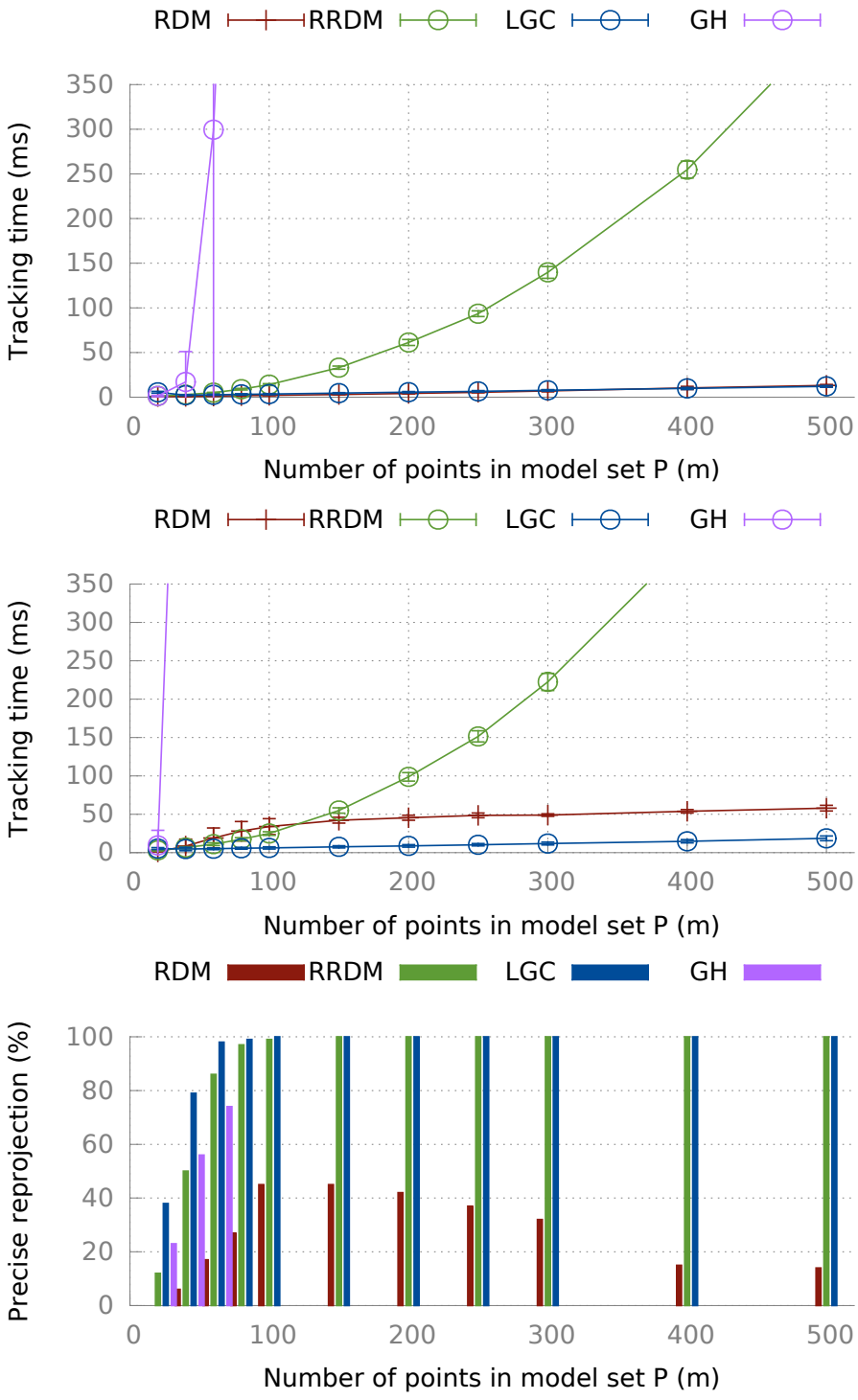


Figure 4.9: Speed experiment. Top: Ideal conditions without jitter nor extra/missing points. Reprojections for all methods are 100% precise. Middle and Bottom: β=15%, η=3%, missing=0%, occlusion=0%.

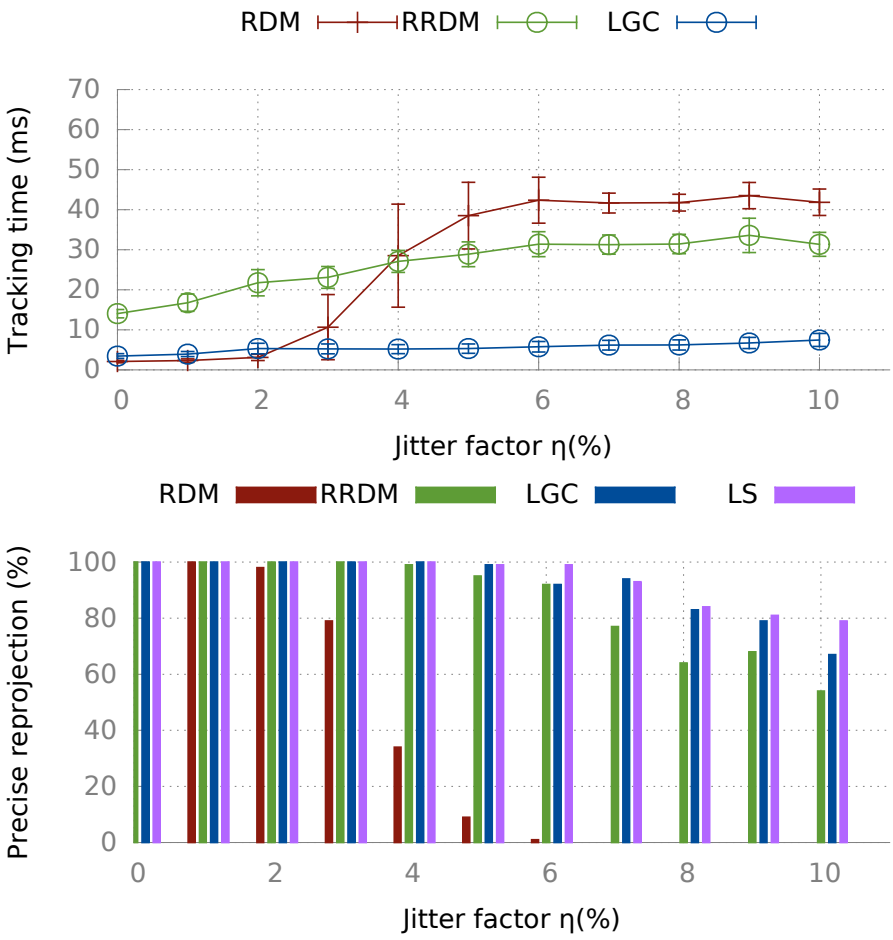


Figure 4.10: Jitter experiment (model P contains 100 points). β=0%, missing=0%, occlusion=0%.

Fig. 4.10 shows results of robustness studies. We increase the jitter factor η to study robustness to acquisition noise which can be due to image processing artifacts, etc. Here, we stick to point sets containing 100 points with no extra or missing points. Clearly, RDM can handle some noise, but is outperformed by both RRDM and LGC. LGC is the fastest and most robust. LS represents results directly found with pre-known point correspondences, which can be seen as the result obtained using ground-truth point correspondences. Note that $\eta = 10\%$ is about the limit of obtaining a precise matching even with ground-truth point correspondences for jittered point pattern matching. LGC sometimes outperforms LS (cf. $\eta = 7\%$ and 8%), because LGC does not keep correspondences when the reprojection distance between the scene and the model points is greater than 3σ (cf. Alg. 8). As a consequence, LGC may use less noisy point correspondences than LS thus leading

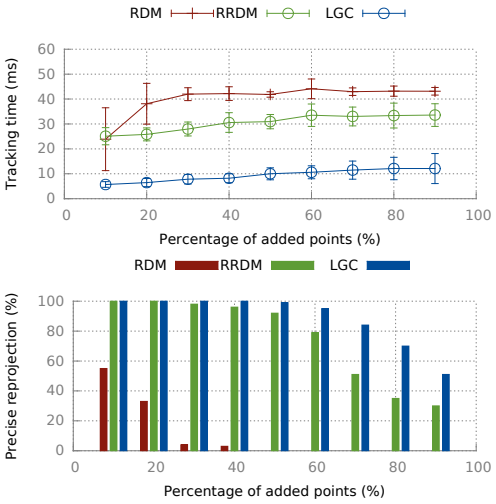


Figure 4.11: Extra points experiment (model P contains 100 points). η=3%, missing=0%, occlusion=0%.

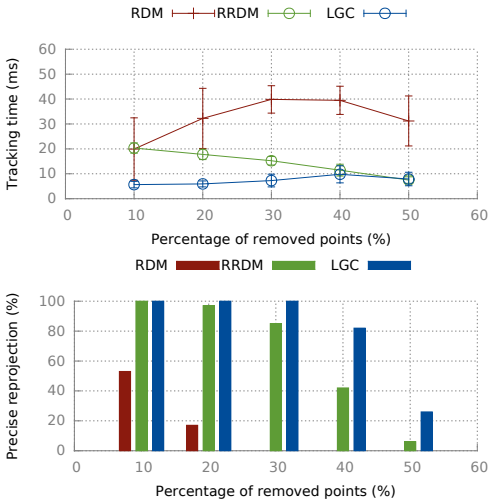


Figure 4.12: Random missing points experiment (model P contains 100 points). β=0%, η=3%, occlusion=0%.

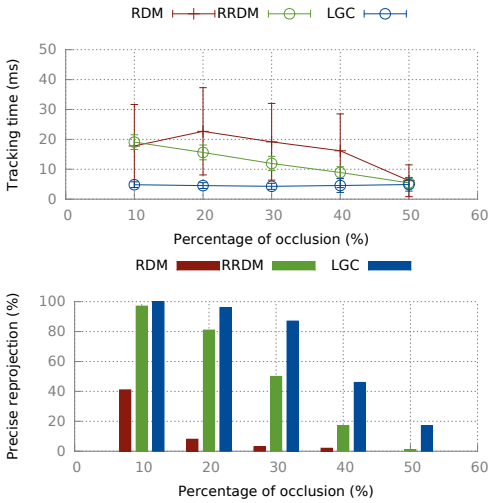


Figure 4.13: Occlusion experiment (model P contains 100 points). β=0%, η=3%, missing=0%, perspective=30°.

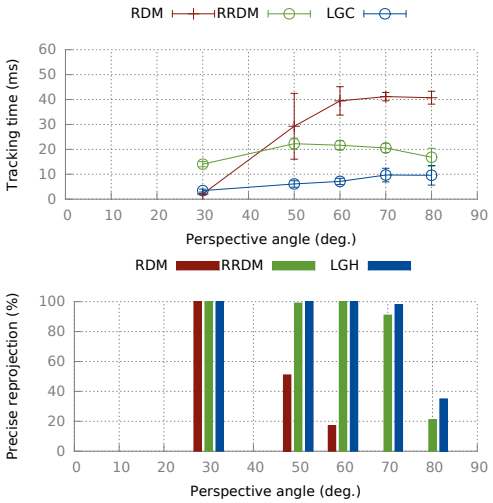


Figure 4.14: Perspective experiment (model P contains 100 points). β=0%, η=3%, missing=0%, occlusion=0%.

to a more accurate homography.

Then, we present results with extra points (such as outliers that could be detected by image processing techniques) in Fig. 4.11, results with randomly removed points (which simulates underdetections) in Fig. 4.12, results when points in a region are masked (i.e. occlusion) in Fig. 4.13 and results with respect to the perspective angle of the camera in Fig. 4.14. In every case LGC is the fastest and the most robust.

At last, Fig. 4.15 shows the discriminative capabilities of both RDM and LGC (RRDM works for only one model). RDM is a bit faster when no jitter occurs while both techniques fully discriminate between several models. RDM fails to retrieve the correct model as soon as jitter is involved. LGC can almost always retrieve the correct one whereas its capability to find an accurate transformation remains the same as in the jitter experiment (Fig. 4.10).

As a conclusion to this theoretical study, our proposal outperforms GH, RDM and our previous work RRDM for both robustness and speed, showing a linear behavior in terms of computation time, whatever the conditions. It has also a great discriminative capability even with large jitter. Next we show that LGC can also be applied to 3D transforms.

4.3.2 3D model registration

We show in this section that our method implemented in 3D, allows for registering models undergoing similarity transformations. Most 3D models are composed of 3D point clouds, forming small surfaces which represent their envelopes. Such point clouds are often obtained from 3D scanners, so they are very dense. As a consequence, the geometric distribution of neighbor points does not contain enough discriminative information for LGC.

We use 3D interest point detectors to find geometrical keypoints in such point clouds. The position of each interest point generally depends on the local topology of the 3D model. So, the spatial distribution of these interest points is more or less random on complex models, which is in favor of LGC's requirements. We perform 3D model registrations by matching detected interest points from these models under similarity transformations. Please remember that we only use point coordinates for matching, neither color information nor normal direction is used.

For illustration purpose, we use the Stanford Bunny with Zhong's Intrinsic Shape Signatures [Zhong 2009] (ISS) since it has a good repeatability [Tombari 2013b, Filipe 2014]. We first create the model point set with ISS key points: for the 35947

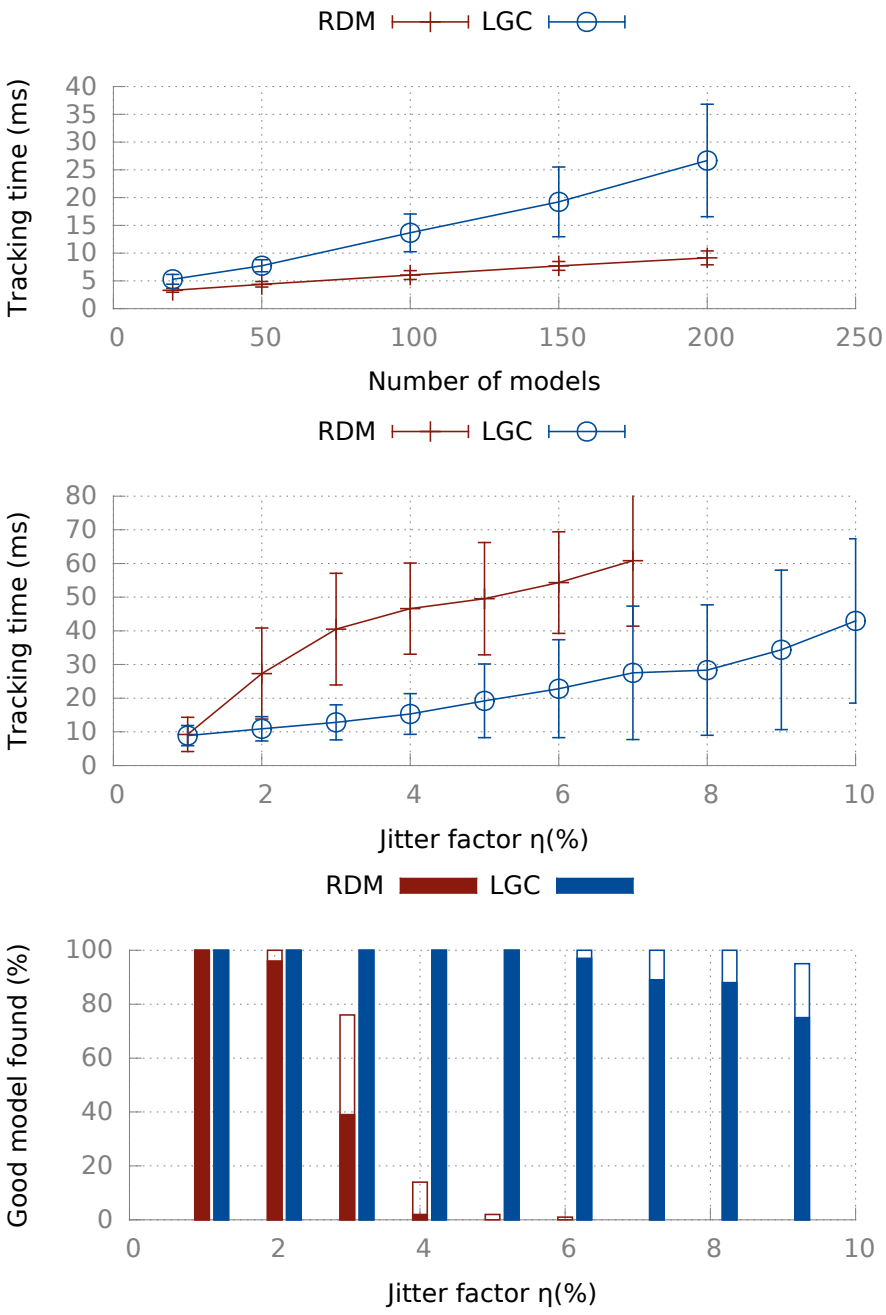


Figure 4.15: Discriminative power. Top: with different number of models. Each model contains 100 points. Bottom: with different jitter factor. 50 models are used with 100 points in each model. Full boxes stand for success, empty boxes for good model found with wrong or imprecise transform. (3.19) is used to judge if a transform is precise.

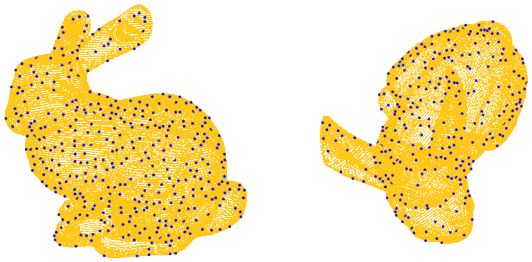


Figure 4.16: Left: the model point set (in blue) detected from the original bunny. Right: an example of scene point set (in blue) detected from a transformed instance.

Table 4.1: Registration results

Instance ID	1	2	3	4	5
Key point size	485	485	479	488	491
Inliers found	459	455	447	457	439
Rotation difference (°)	0.10	0.06	0.00	0.10	0.10
Scale difference (%)	0.09	0.04	0.05	0.05	0.09

Bunny points, we obtain 480 keypoints. Then, random similarity transformations are applied on the bunny model to create 5 instances for matching (cf. Fig. 4.16). η is set to 3%. Results of registration are listed in Table 4.1. In all cases, rotation errors are less than 0.1° while scale errors are less than 0.1%. We can conclude that LGC has the potential to be also used for 3D similarities estimation.

4.3.3 LGC with additional information

So far, we have only considered pure point matching problems where each point is indistinguishable from another. However, points may have their own properties including color or local feature in textures. We show here that such properties can be used in our algorithm.

Some traditional feature point matching methods (e.g. SIFT, SURF or BRIEF) are based on these different properties of points. They first find rough correspondences by using point properties before applying RANSAC-like methods. Since such methods need high inlier ratios, the rough correspondences should contain a lot of inliers. This implies that point properties should be discriminant enough, which results in high dimensional descriptors. We show that using our method instead of RANSAC greatly reduces the descriptors' dimension.

In our experiment we use ORB [Rublee 2011]. The original ORB descriptor is composed of a sequence of 32 bits generated from random pixel comparisons. As a consequence, it is very easy to get lower dimension descriptors from the original ones. Image points are no longer fed to the generator randomly but according to their nearest Hamming distances to model points. Let the length of the descriptor in bits be L_{ORB}. In the generator, we do not process (p_j-patch, q-patch) which Hamming distance is larger than $L_{ORB}/2$ in Alg. 6, thus reducing the number of false hypotheses. The condition in Alg. 8 is also relaxed to find more corresponding points: we add (p', q') to the solution list only if the Hamming distance between p' and q' is less than $L_{ORB}/3$ and if $|p' - T^{-1}(q')| \leq 3d_t$. We use these values for illustration purpose, they are not optimized.

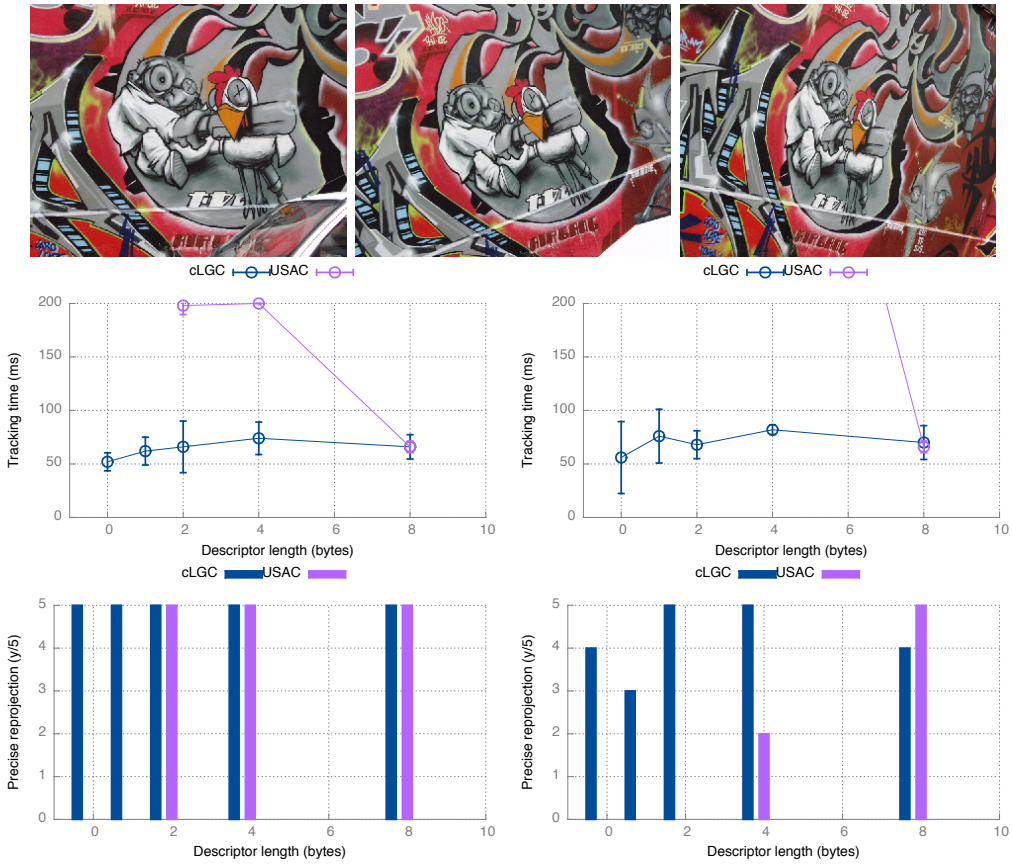


Figure 4.17: Graf series and results. Top: graf figures. Bottom left: result for image pair (left, middle). Bottom right: result for image pair (left, right).

The original 32 bytes ORB are chopped to have lower dimensional descriptors which are used to compare our method with USAC [Raguram 2013a]. The experiment is repeated 5 times for each case. Results are presented in Fig. 4.17 where we

can see that LGC works with much smaller descriptors and is faster than USAC.

4.4 Applications

4.4.1 Tracking ordinary planar objects

Corners in real textured images are assumed to be randomly distributed. As demonstrated in [Uchiyama 2011a], these corners can be used for tracking. We compare our method with RDM and SURF in this section on real planar targets. However, our purpose is not to compete with textural-based trackers in these situations. SURF serves as a reference. We aim at illustrating the flexibility of our approach to deal with traditional targets and showing its robustness on real targets.

We use the OpenCV "goodFeaturesToTrack" method [Shi 1994] as interest point detector in this experiment which performs at about 10ms/frame. The *maxCorners* and *minDistance* parameters are selected in favor of RDM according to [Uchiyama 2011a] (*qualityLevel*=0.13 and *blockSize*=12 for stable point detection). All remaining parameters are set to their default values. As to SURF, a FLANN based matcher in OpenCV is used to build a database of descriptors for tracking.

We use a Logitech C270 camera with resolution set to 640×480. The experiment is as follows: first eight different coplanar objects (cf. Fig. 4.18) pass individually in front of the camera. The same top view of each object is registered using three algorithms (RDM, LGC and SURF). Then, each object is tracked by the three algorithms separately.

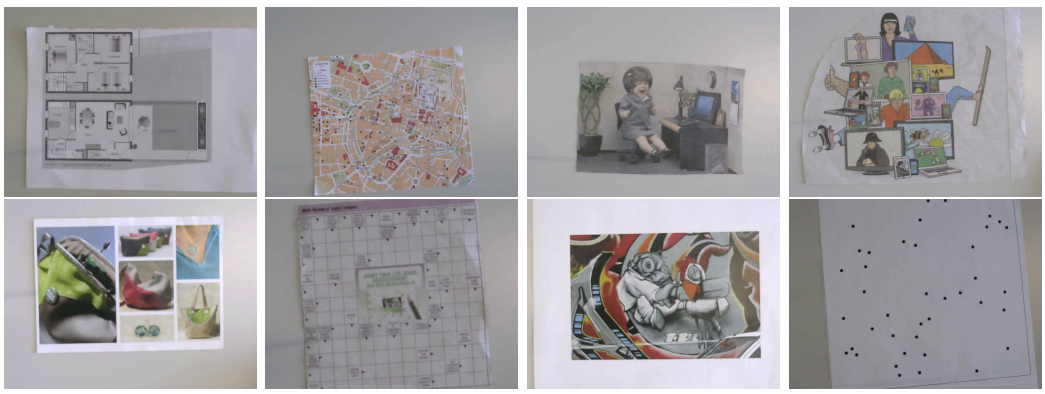


Figure 4.18: Models used in this experiment. From left to right: (Top) appartment-plan, city-map, person, advertisement; (Bottom) postcard, cross-word, graffiti, point-pattern.

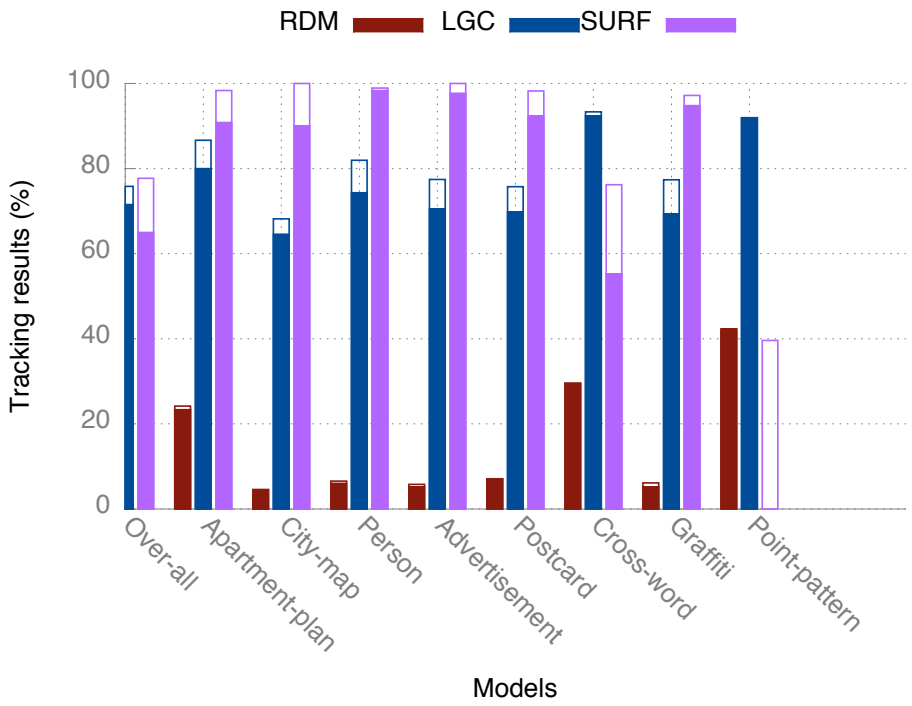


Figure 4.19: Tracking results: Solid bars stand for correct matching, empty bars for bad matching (i.e. the algorithm matches the scene to a wrong model, or produces an imprecise matching by visual assessments).

Similar experiments are repeated twice for a total of 1753 frames and manual visual assessments are provided for all estimated homographies. Both SURF and LGC match at about 15-18ms/frame while RDM matches at around 50ms/frame. Fig. 4.19 shows a summary of the results for each model during tracking. LGC outperforms RDM in all cases. Although SURF performs better than LGC in general, it works badly on cross-word and point-pattern since these two models contain less textures. As RDM can be used for "augmenting everything" [Uchiyama 2011a], we claim our method is a better alternative to RDM for this purpose.

4.4.2 Augmenting engineering drawings

Engineering drawings are largely used in mechanical engineering and architectural design. They are different from ordinary texture-rich models because they are most of the time 2D representations that contain only geometric information (such as straight lines and circles) of 3D CAD models. Traditional texture-based keypoint tracking methods cannot deal with such cases. Fig. 4.20 presents results of augmenting engineering drawings with their 3D models. We extract in real-time the

intersections of the drawing which are mapped to previously created model point sets.

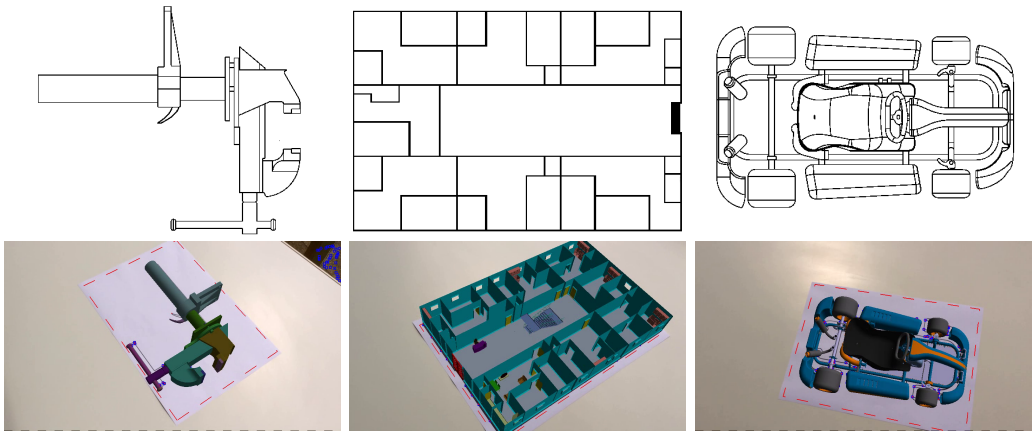


Figure 4.20: Augmenting CAD drawings of a ragum, an apartment and a kart (see supplementary video).

4.5 Discussion

In this section, we discuss some aspects of the algorithm.

4.5.1 Neighbors

We use nearest neighbors to create local patches but use mesh neighbors in the refiner. We tried to use the same type of neighbors in the algorithm but neither performs better. Although mesh neighbors are more robust against perspective distortions [McIlroy 2012], they are very sensitive to extra/missing points, leading to a less robust generator. On the other hand, when points are gathered into two or more local groups, nearest neighbors may give rise to "important edges" in the resulting network (cf. Fig. 4.21). If these edges disappear due to extra/missing points, two or more subregions are disconnected. Thus by visiting nearest neighbors in the refiner, inliers are often confined to a small subregion, which gives an imprecise estimation of the transformation.

4.5.2 Transformation T

Theoretically, the algorithm works with any transformation T, but some conditions on point sets have to be satisfied.

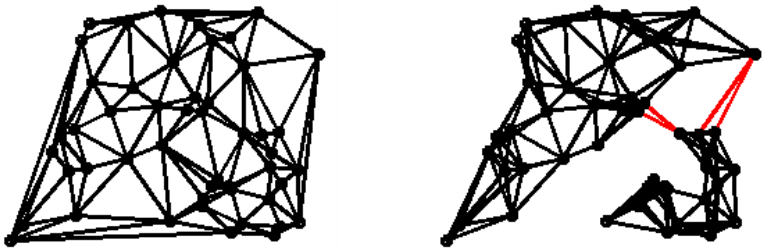


Figure 4.21: Delaunay mesh neighbors (left) and nearest neighbors (right) of the same point set. Neighboring points (black dots) are connected by a black edge. Mesh neighbors connect points better than nearest neighbors, the latter giving rise to four "important edges" represented in red.

Let $\mathbf{u_0}$ and $\mathbf{u}$ be two points which may not belong to point patterns. $\mathbf{u}$ is in the neighborhood of $\mathbf{u_0}$. We assume that T can be expanded by the Taylor theorem in the neighborhood of $\mathbf{u_0}$:

$$T(\mathbf{u}) = T(\mathbf{u_0}) + \frac{\partial T(\mathbf{u_0})}{\partial \mathbf{u}}\Delta\mathbf{u} + \frac{\partial^2 T(\mathbf{u_0}+\lambda\Delta\mathbf{u})}{\partial \mathbf{u}^2}(\Delta\mathbf{u})^2 \tag{4.17}$$

with $\Delta\mathbf{u} = \mathbf{u}-\mathbf{u_0}$ and $\lambda \in [0,1]$. The first two terms on the right are the linear local transformation $T_L(\mathbf{u})$ that we mentioned before, the last term is a small quantity of the same order as $(\Delta\mathbf{u})^2$, which represents the difference between $T(\mathbf{u})$ and $T_L(\mathbf{u})$.

If the difference between $T(\mathbf{u})$ and $T_L(\mathbf{u})$ is not very big, the last term can be easily managed by considering that it represents the "jitter". The difference can be expressed as:

$$\|T(\mathbf{u}) - T_L(\mathbf{u})\| \leq \left\| \frac{\partial^2 T(\mathbf{u_0}+\lambda\Delta\mathbf{u})}{\partial \mathbf{u}^2} \right\| l^2 \tag{4.18}$$

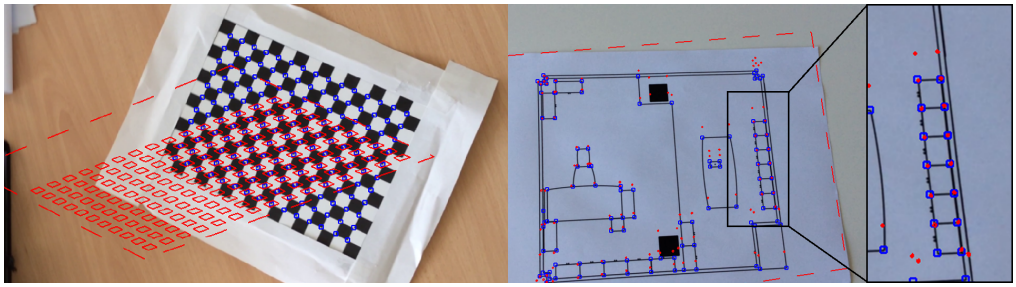
$\|\frac{\partial^2 T(\mathbf{u_0}+\lambda\Delta\mathbf{u})}{\partial \mathbf{u}^2}\|$ depends only on transformation T, not on inter-point distance l. Recall that $0.1l$ is the limit of algorithm in terms of jitter (cf. Section 4.3). If the inter-point distance l is small enough so that $l \leq 0.1\|\frac{\partial^2 T(\mathbf{u_0}+\lambda\Delta\mathbf{u})}{\partial \mathbf{u}^2}\|^{-1}$, then we have:

$$\|T(\mathbf{u}) - T_L(\mathbf{u})\| \leq \left\| \frac{\partial^2 T(\mathbf{u_0}+\lambda\Delta\mathbf{u})}{\partial \mathbf{u}^2} \right\| l^2 \leq 0.1l \tag{4.19}$$

The difference between $T(\mathbf{u})$ and $T_L(\mathbf{u})$ can be smaller than $0.1l$. So the algorithm can theoretically deal with any transformation if the distribution of points is dense enough.

4.5.3 Repetitive structures

As our method relies on discriminating local geometric structures, repetitive patterns are an issue. They impact both the "generator" and the "validator" since they work on local patterns, but do not affect the "refiner". If the hypothesis (cf. Sec. 4.1.3) is generated inside a repetitive pattern, the algorithm will probably give a false result; otherwise both "generator" and "validator" have enough discriminative information to estimate a true correspondence. As the generator selects scene points randomly, the algorithm's performance and the percentage of points in repetitive patterns are in negative correlation. Our experiments show that the algorithm fails with exclusively regular patterns such as a chessboard (cf. Fig. 4.22a), but can successfully handle small repetitive structures inside a more global irregular one, indeed only 2 frames out of 317 fail in the "office design" case (cf. Fig. 4.22b). In the latter example, we rely on [Pham 2014] to extract intersections and junctions.



(a) (b)

Figure 4.22: Impact of regular patterns. (a) chessboard, 165/165 repetitive feature points. (b) office design, 26/109 repetitive feature points. Blue points are detected while red points are projected model points.

4.6 Conclusion

In this chapter, we have presented LGC, an algorithm based on Local Geometric Consensus that can be used for several transformation types, both in 2D and 3D.

Compared to RRDM in Chapter 3 and RDM [Uchiyama 2011b], it is more generic and can be applied to any known 2D/3D transformation. It has the ability to recognize and match several different models, which cannot be solved by RRDM. As to the case of 2D homography, LGC shows a linear time complexity behavior, which is much faster and more robust than RRDM (quadratic time complexity). LGC is much more robust than RDM [Uchiyama 2011b] in a noisy circumstance.

We have also shown two real-time applications developed by using LGC tracking. It concludes that wide baseline point pattern matching performed by LGC can be both fast and robust enough for texture-less augmented reality applications. In the following chapter, we will show one more application by applying LGC.

Chapter 5

Defocused projector calibration for projector-camera systems

Contents

Projectors are important display devices for large scale augmented reality applications. Calibration is one of the most essential elements of projector-camera systems (referred to as ProCam systems in the following), namely determining the intrinsic matrix and the distortion coefficients of cameras and projectors, as well as their relative position. A precise calibration will produce a precise mapping between

the physical world and camera/projector images, which permits the augmented information to be projected to their correct positions in SAR. However, precisely calibrating projectors with large focus distances implies a trade-off between practicality and accuracy. People either need a huge calibration board or a precise 3D model [Resch 2015].

With the help of LGC introduced in Chapter 4, we present a practical ProCam calibration method to solve this problem. The user only needs a small calibration board to calibrate the system regardless of the focus distance of the projector. Results show that the Root-Mean-Squared reprojection Error (RMSE) for a 450cm projection distance are only about 4mm, even though it is calibrated using a small B4 (250×353mm) calibration board. This work has been presented at *International Symposium on Mixed and Augmented Reality (ISMAR)* in 2016.

5.1 Related work

There exist two families of methods for projector calibration: (i) calibrating each light stripe (cf. Fig. 5.2) [Yamauchi 2008] or each pixel [Luo 2014] of the projector, and (ii) calibrating the projector as a whole using the pinhole model. The former is very time consuming since each light stripe or pixel needs to be calibrated individually. The latter mainly features two categories of techniques: two-views based methods and inverse camera methods.

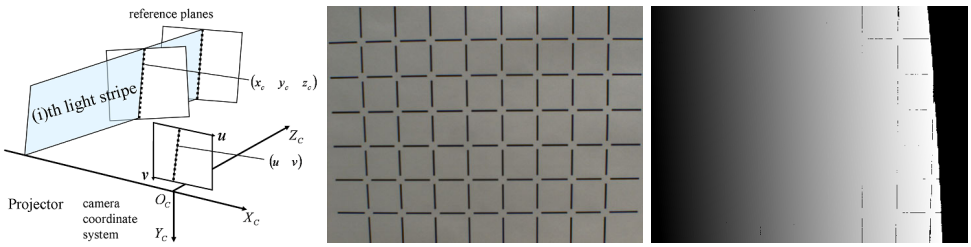


Figure 5.1: Projector's light stripe calibration. Left: the setup of the calibration system. Middle: calibration board. Right: light stripe projected onto the calibration board. [Yamauchi 2008]

5.1.1 Two-views based methods

As an example of "two-views based methods", Yamazaki et al. [Yamazaki 2011] use structured-light patterns to create dense point correspondences between the camera and the projector views to find the fundamental matrix by exploiting two-view

geometry properties. The final intrinsic matrices of both the projector and the camera are calculated iteratively by assuming their initial values (by referring to the reference manual for example [Yamazaki 2011]). This method has no error propagation but the matrices of the projector and of the camera have a coupling-effect. It is sensitive to initial values and only estimates 3 parameters of the intrinsic matrix. With the help of a known precise 3D model of a target object, Resch et al. [Resch 2015] use bundle adjustment to iteratively correct the estimated matrices without dependence on initial values. However, a precise 3D model is not always available in the general case.

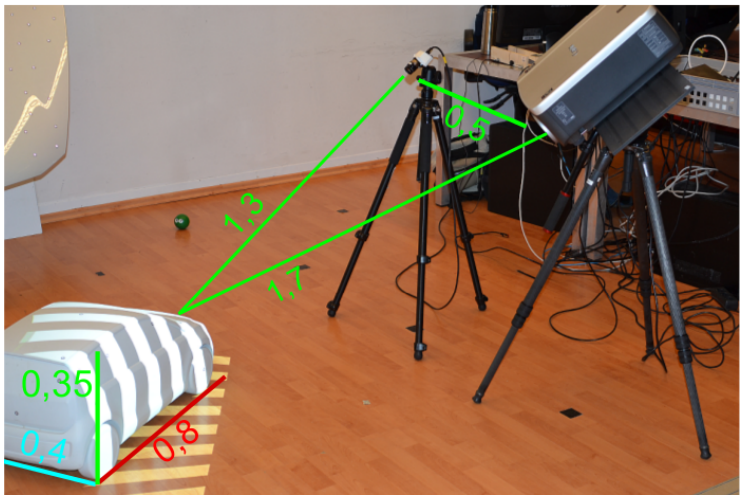


Figure 5.2: An example of setup of two-views based method. The car and its corresponding 3D numeric model is the calibration object. Structured-light patterns are projected onto the car [Resch 2015].

5.1.2 Inverse camera methods

This kind of methods treats projectors as inverse cameras, so that they can be calibrated using Zhang's method [Zhang 2000]. More precisely, a calibration board with a pre-defined pattern is used to calibrate the camera. In the meantime, the projector projects other patterns onto the calibration board. These projected patterns are used to find camera-projector point correspondences in order to calibrate the projector. Several methods exist for finding such correspondences. Structured-light methods [Zhang 2006, Li 2008] employ projections of series of coded gray-bars (cf. Fig. 5.3a). Even though methods for defocused projector exists [Li 2014], the board needs to be frozen at the same position by the user for several seconds, which is not convenient. Other methods capture fewer or even a single image for each board

position, e.g. by projecting regular dot patterns (cf. Fig. 5.3b) [Ouellet 2008], a chessboard (cf. Fig. 5.3c [Gao 2008]) or a matrix of ARToolKit markers (cf. Fig. 5.3d) [Audet 2009]. Our method follows a similar approach.

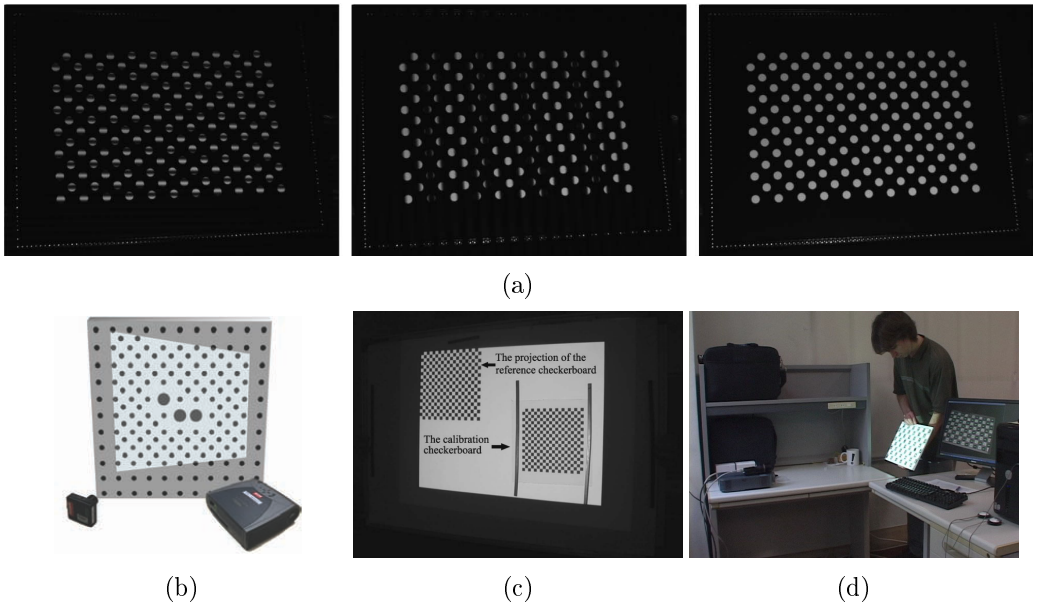


Figure 5.3: Different calibration pattern used in inverse camera methods. (a) Regular dot pattern is used and point correspondences between projector and camera are found by projecting structured-lights [Li 2014]. (b) A physical and a projected regular dot pattern is used [Ouellet 2008]. (c) A physical and a projected chessboard is used [Gao 2008]. (d) Two matrix of ARToolKit markers are used [Audet 2009].

5.1.3 Limitations

In [Gao 2008] and [Audet 2009], the calibration board has to be positioned in the focus zone (where projected images are clear), or the pattern becomes too blurred to be recognized. Moreover, Zhang's method [Zhang 2000] requires the pattern to occupy a large part of the field of view of the projector in order to give a better result. Consequently if a wide-angle projector is focusing at about 2m, the user needs to manipulate at least an A0 size (841×1189mm) calibration board, which is a burden. The user can also choose to calibrate the projector at a short distance and then change the focus depending on the application, but this will induce a loss of precision, as showed later in Section 5.5. We also experimentally noticed that for Audet et al.'s method [Audet 2009], projectors' brightness greatly influences markers recognition.

Ouellet et al. [Ouellet 2008] need to take 3 images for each board position while the projector is successively projecting (1) a white image, (2) a regular pattern, (3) a calibration regular pattern. These images are highly related: (1) is subtracted from (2) to find the homography between the projector and the camera (H_{cp}). If the board moves or ambient lighting changes, detection may fail. (3) needs H_{cp} to interleave projected points between printed ones. If the board moves, points may interfere. Moreover, a few missing points in the regular pattern can lead to detection failures. Thus the whole process is not very robust. At last, this method detects projected dot centers by back projecting them onto the board plane. When the board is out of focus, back projected dots may be heavily deformed, which can make the center detection difficult and introduce large errors.

5.1.4 Contribution

Compared to [Audet 2009] and [Ouellet 2008], our work has the following advantages which makes it much more practical and usable:

- It can work at large distances without having to manipulate unpractical huge calibration boards, unlike [Audet 2009]
- It uses random dot patterns, which are robust to pattern interference and insensitive to lighting, unlike both [Audet 2009] and [Ouellet 2008]
- It only needs one image for each board position, thus has a faster recovery from detection failure, unlike [Ouellet 2008]
- It gives more stable results for intrinsic matrices of cameras and projectors, compared to [Audet 2009]

5.2 Calibration Method

Our method relies on simple manipulations of a calibration board whatever the projector's focus distance is: the user holds a small calibration board before the camera and the projector. If the board is still for a while ($\approx$ 1s), an image is automatically captured for calibration. Once a pre-defined number (K, cf. Section 5.4) of images are acquired, the system can be calibrated using Zhang's method [Zhang 2000] (cf. Fig. 5.4).

In the following, we first introduce some basic notations and the generic "inverse camera" method. Then we present our calibration patterns and how to find feature correspondences. Finally, the whole calibration procedure is presented.

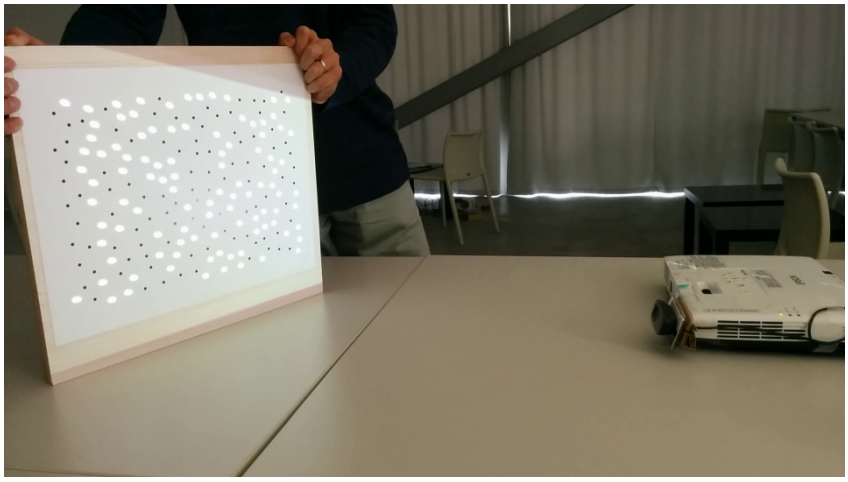


Figure 5.4: Calibration: only a small board is manipulated whatever the focus distance of the projector.

5.2.1 Basic notations and inverse camera method

The ProCam system is described by the camera's intrinsic matrix K_c, its distortion coefficients D_c, the projector's intrinsic matrix K_p, its distortion coefficients D_p, the projector rotation in camera frame R and the position of the projector's optical center in camera frame t. More precisely, the intrinsic matrix is defined by [Bradski 2008] as:

$$\mathbf{K}_\varepsilon = \begin{pmatrix} f_{\varepsilon x} & \gamma_\varepsilon & u_\varepsilon \\ 0 & f_{\varepsilon y} & v_\varepsilon \\ 0 & 0 & 1 \end{pmatrix} \tag{5.1}$$

where ε stands for c or p, $f_{\varepsilon x}$ and $f_{\varepsilon y}$ are focal lengths, $(u_\varepsilon, v_\varepsilon)$ is the position of the principal point, γ_ε is the skew.

The "inverse camera" method starts with two sets of correspondences for each view of the calibration board: $c_{cb} = \{(\mathbf{x}, \mathbf{x}^{(c)})\}$ and $c_{cp} = \{(\mathbf{y}^{(p)}, \mathbf{y}^{(c)})\}$, where $\mathbf{x}$ is an interest point (corner/center of markers, etc.) on the calibration board, $\mathbf{x}^{(c)}$ is its image in the camera; $\mathbf{y}^{(p)}$ is an interest point in the projected image, $\mathbf{y}^{(c)}$ is its image viewed by camera. With c_{cb} obtained from $K (K \geq 3)$ views, the camera can be first calibrated using Zhang's method [Zhang 2000]. Lens distortions in $\mathbf{x}^{(c)}$ and $\mathbf{y}^{(c)}$ can then be removed.

For the projector, the homography from the board to the camera H_{cb} is found with c_{cb}. The location on the calibration board of projected features $\mathbf{y}^{(p)}$ can then be expressed as $H_{cb}^{-1}\mathbf{y}^{(c)}$, thus a new correspondence set $c_{pb} = \{(\mathbf{y}^{(p)}, H_{cb}^{-1}\mathbf{y}^{(c)})\}$

can be established for each view. At last, the projector can be calibrated with c_{pb} obtained from K views.

5.2.2 Calibration pattern

We choose to use randomly distributed circular dots as both physical patterns printed on the calibration board and projected patterns, with the following advantages:

- Circular points have no internal structures unlike ARToolKit markers, they are thus less influenced by defocus or blur or lighting changes.
- Different random dots-based markers can be distinguished, real-time tracked and accurately located by LGC even in the case of partial occlusion and over/under-detections, unlike regular patterns. This makes the method robust against pattern overlapping (cf. Chapter 4).
- Points occupy a small surface: the same area can contain more points than other geometries, so more correspondences can be established

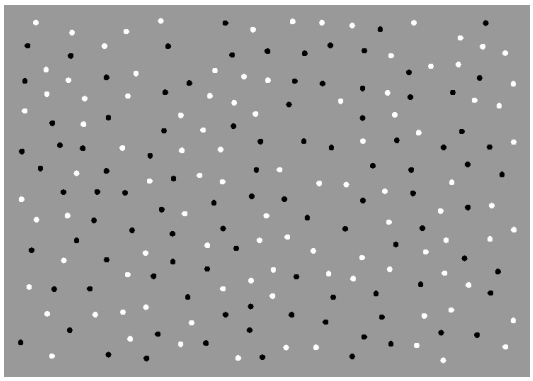


Figure 5.5: Calibration Patterns: P_b in black are printed on a piece of paper. P_p in white are projected. Both form the original pattern P_o.

The calibration pattern (cf. Fig. 5.5) is constructed as follows: $2N$ points are randomly generated in a rectangle region, called the original pattern (P_o) afterwards. The first half of these points (P_b) are printed on a paper and attached to a rigid board, the second half (P_p) is pre-warped and projected onto the board in order to form P_o during the calibration procedure (cf. Section 5.2.3). We set a minimum inter-point distance for all $2N$ points to have a more homogeneous distribution, which improves camera calibration according to [Shih 1996]. Since our objective

is to calibrate the projector with a far focus by using a small calibration board, the board does not have to be close to the plane of focus of the projector, so the projected points will be defocused and become larger. The minimum inter-point distance requirement makes the defocused projected points less likely to overlap with printed points.

5.2.3 Algorithm

We use LGC to find point correspondences, the two basic operations of this algorithm are defined as follows:

$$LGC.register(P) \tag{5.2}$$

$$[\tilde{P}, P_i, H] = LGC.match(I) \tag{5.3}$$

(5.2) registers a model point pattern P into LGC, several models may be registered; (5.3) matches detected black points in image I with all registered models. If a model P_i is found in the detected pattern, (5.3) returns P_i and its corresponding detected point set $\tilde{P}$. An homography $H : \tilde{P} = H(P_i)$ is also returned.

The algorithm contains three parts: initialization, manipulation and improvements (cf. Alg. 9). For initialization, both P_b and P_p points are registered into LGC as models. An initial value H_{pre} is defined so that $H_{pre}(P_p)$ exactly fits the projector resolution.

During manipulation, the algorithm matches P_b (i.e. the calibration board) with LGC. Once the board is found, it starts to detect and match P_p (i.e. the projected pattern) on the calibration board in the same way. If P_p is found as well, an homography H_{pre} is computed whose role is to warp the projected pattern to the right position for the next iteration (cf. Fig. 5.6). Note that H_{pre} only warps points' positions, while the points to be projected are always circular. When both patterns are well *aligned* (meaning that they form well P_o) and the board is *steady* for $\approx$ 1s, an image I is captured, and both H_{bc} and H_{pc} are recorded. We choose $t_s = 3.0$ pixels and $t_a = 2.0$ pixels in Alg. 9 empirically.

Once images of K views are obtained, an improvement step is executed. It is used to deal with the fact that the center of a circle is not perspective invariant: after applying a perspective transformation, the transformed center of the circle is not the center of the transformed circle. The approach is similar to [Datta 2009]: each captured image I is rectified so that points are nearly circle. More precisely, since the transformation from the calibration board to the camera image plane is

the homography H_{cb} because of pinhole camera, rectified views $H_{cb}^{-1}(I)$ are used for physical points P'_b. As we use the pinhole projector model, the relationship between the projector image and the camera image is the homography H_{cp} induced by the calibration board [Hartley 2004]. Therefore, rectified views $H_{cp}^{-1}(I)$ are used for projected points P'_p. This latter approach introduces the defocusing error, which is discussed in the following section. To detect ellipse centers in rectified views, the OpenCV's MSER detector [Nistér 2008] is used with its default parameters.

Algorithm 9 Algorithm

$LGC.register(P_b)$
$LGC.register(P_p)$
set H_{pre} such that $H_{pre}(P_p)$ exactly fits the projector's resolution
$k = 0$
while $k < K$ **do**
  Projector: Draw circles at $H_{pre}(P_p)$ and project them
  Camera: Get new image I
  $[P'_b, P_b, H_{cb}] = LGC.match(I)$
  if P'_b found **then**
    $[P'_p, P_p, H_{cp}] = LGC.match(\tilde{I})$, $\tilde{I}$ is I with inverted color.
    if P'_p found **then**
      $steady = \forall p \in P'_b$ has moved less than t_s pixels since last image
      $aligned = \forall p \in P_b, \|H_{cb}(p) - H_{cp}(p)\| < t_a$ pixels
      if *steady* && *aligned* **then**
        if last over $1s$ **then**
          Capture I, record correspondences (P'_b, P_b), $(P'_p, H_{pre}(P_p))$ and homographies H_{cb}, H_{cp}
          $k++$
      else
        $H_{pre} = H_{pre} H_{cp}^{-1} H_{cb}$

Improve detection of P'_b in rectified views $I_b = H_{cb}^{-1} I$
Improve detection of P'_p in rectified views $I_p = H_{cp}^{-1} I$
Calibrate the system (cf. Section 5.2.1)

The initialization step takes $\approx$ 1s, the manipulation sequence is performed in real-time, the improvement step can process $2-3$ frames per second.

5.3 Defocusing error

As mentioned in Section 5.2.3, the center of a circle is not perspective invariant. This can be dealt with view rectification. But if the transformed circle is defocused, the view rectification cannot compensate the error introduced by the defocus effect.

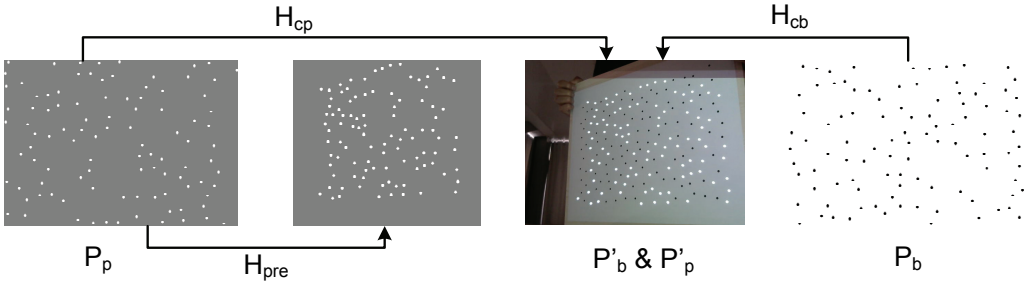


Figure 5.6: Point patterns used in Alg. 9. From left to right: P_p, projected pattern $H_{pre}(P_p)$, P'_b and P'_p in camera view, board pattern P_b. $H_{cp} = H_{cb}$ if P'_b and P'_p are well aligned on board.

In this section, we study the error in center localization by defocus.

5.3.1 The origin of the defocusing error

An optical schema of a single point projection is shown in Fig. 5.7. Since the depth of field of pinhole cameras/projectors is infinite, a projected dot is never defocused despite of the distance between the calibration board and the projector (cf. the green cone in Fig. 5.7). But when the light follows a lens model inside the projector (cf. the blue cone in Fig. 5.7), there is only one plane of focus. Therefore, a projected dot is defocused if the calibration board is not exactly on the plane of focus. Therefore, in order to estimate the defocusing error, the lens model should be considered.

In Fig. 5.7, the projector lens is located at O. A circle $\mathbf{s}_o$ is located on the projector's plane p_o. It is projected and forms a clear image $\mathbf{s}_i$ on the plane of focus p_i. All light coming from $\mathbf{s}_o$ and being transmitted by the lens form a blue cone C_b, which follows a standard lens projection model. C_b intersects the board plane p_b and forms the light spot $\mathbf{s}_b$ on the board.

During the calibration, $\mathbf{s}_b$ is captured by the camera and forms an image $H_{cb}(\mathbf{s}_b)$ on the camera's image plane. To detect the centers of $\mathbf{s}_b$, homography H_{cp}^{-1} is used to rectify the image $H_{cb}(\mathbf{s}_b)$. This process implies that a pinhole projector model is used (cf. Section 5.2.3). The rectified spot is $H_{cp}^{-1}H_{cb}(\mathbf{s}_b) = H_{pb}(\mathbf{s}_b)$. H_{pb} is the homography from positions on the calibration board p_b to those on the projector image plane p_o induced by a pinhole projector model. As a consequence, if we think that the projector works inversely, then the rectified $H_{pb}(\mathbf{s}_b)$ can be regarded as an image of $\mathbf{s}_b$ "seen" by the pinhole inverse-projector. In Fig. 5.7, light from $\mathbf{s}_b$ following the pinhole inverse-projector model is presented by a red elliptic cone and thus $\mathbf{s}_r = H_{pb}(\mathbf{s}_b)$ is the rectified spot. The MSER detector detects the center of $\mathbf{s}_r$,

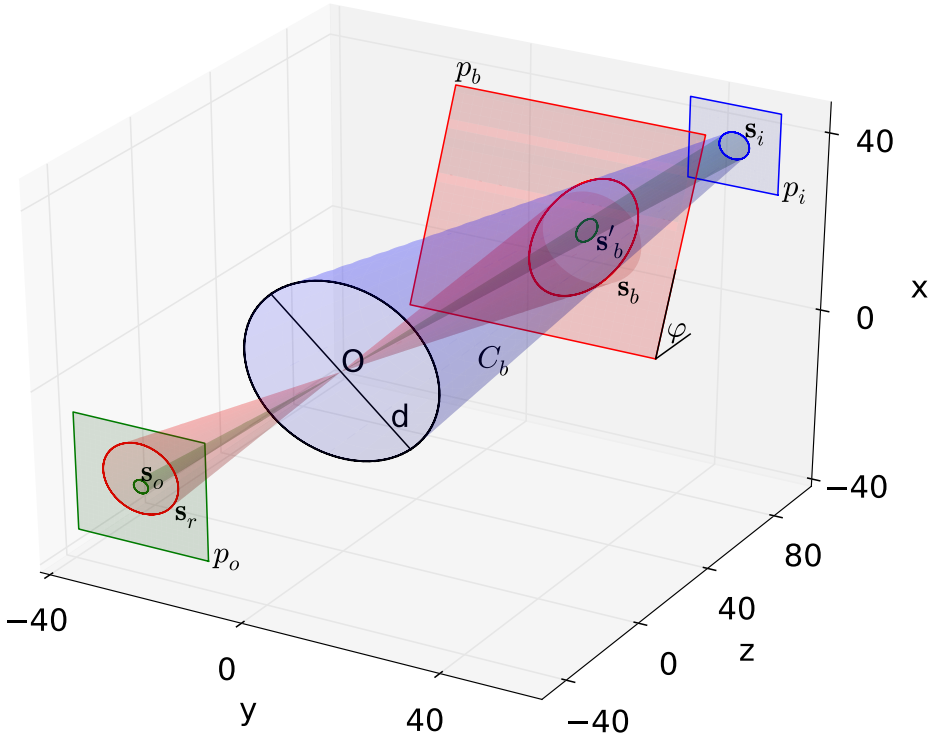


Figure 5.7: Projection geometry for defocusing error: O is the projector optical center, p_o is the projector plane, p_i is the plane of focus, p_b is the board plane. The blue cone represents the light that follows a lens projector model. $\mathbf{s}_o$ (green) is projected by the lens and forms a defocused spot $\mathbf{s}_b$ on the board. Both the green cone and red cone represent the light that follows a pinhole projector model. If $\mathbf{s}_o$ is projected by a pinhole projector (the green cone), it should form a clear spot $\mathbf{s}'_b$ on p_b. $\mathbf{s}_r$ (red) is the defocused spot "seen" by an inverse pinhole projector (the red cone).

denoted as E_r, and we use $(E_o, H_{pb}^{-1}(E_r))$ as one correspondence for the projector calibration.

However, if there is no defocus in the optical system (i.e. with the pinhole projector model), the light from $\mathbf{s}_o$ should form a green cone and produces an image $\mathbf{s}'_b = H_{pb}^{-1}(\mathbf{s}_o)$ on the calibration board p_b. If the center of $\mathbf{s}_o$ is denoted as E_o, then the correspondence $(E_o, H_{pb}(E_o))$ should be used theoretically without any defocus. Unfortunately, $\mathbf{s}'_b$ is physically not observable, so $H_{pb}^{-1}(E_r)$ is used instead of $H_{pb}^{-1}(E_o)$ which introduces a defocusing error $H_{pb}^{-1}(E_r) - H_{pb}^{-1}(E_o)$. The projection of this defocusing error on the projector's object plane p_o is then $e = E_r - E_o$.

5.3.2 An estimation of the defocusing error

To derive the error e, a coordinate system is constructed so that the angle between p_b and the projector's z-axis is φ. Let O be the origin of the coordinate system, as well as the projector's optical center. The z-axis intersects p_b at point $(0, 0, b)$, meaning that b represents the distance between p_b and the projector. Therefore, the equation of the board plane p_b is:

$$p_b : z - x\cot\varphi - b = 0 \tag{5.4}$$

The projector's plane p_o is at $z = z_o$ and the plane of focus p_i is at $z = z_i$. d is the diameter of the projector lens. The circle $\mathbf{s}_o$ has it center at $E_o(x_o, y_o, z_o)$ with r_o, the circle $\mathbf{s}_i$ has a radius of r_i. Then the equation of C_b is:

$$C_b : \begin{cases} x(t,\alpha) &= \frac{x_o}{z_o}t + \frac{d}{2}\left[1 - (1 - \frac{2r_i}{d})\frac{t}{z_i}\right]\cos\alpha \\ y(t,\alpha) &= \frac{y_o}{z_o}t + \frac{d}{2}\left[1 - (1 - \frac{2r_i}{d})\frac{t}{z_i}\right]\sin\alpha \\ z(t,\alpha) &= t \end{cases} \tag{5.5}$$

where $t > 0$, $\alpha \in [0, 2\pi)$ are two parameters for C_b.

By using the fact that $\mathbf{s}_b$ is the intersection of C_b and p_b; both $\mathbf{s}_b$ and $\mathbf{s}_r$ lie on C_r; C_r is an elliptic cone passing through O; and $\mathbf{s}_r$ is the intersection between C_r and p_o. We can find the equation of $\mathbf{s}_r$ with parameter $\alpha \in [0, 2\pi)$:

$$\mathbf{s}_r : \begin{cases} x(\alpha) &= x_o + \frac{z_o d\cos\alpha}{2b + d\cot\varphi\cos\alpha}\left(1 - \frac{x_o}{z_o}\cot\varphi - \frac{b}{z_i} - \frac{2br_o}{z_o d}\right), \\ y(\alpha) &= y_o + \frac{z_o d\sin\alpha}{2b + d\cot\varphi\cos\alpha}\left(1 - \frac{x_o}{z_o}\cot\varphi - \frac{b}{z_i} - \frac{2br_o}{z_o d}\right). \end{cases} \tag{5.6}$$

We can prove that $\mathbf{s}_r$ is an ellipse centered at (x_r, y_r), with:

$$\begin{cases} x_r &= x_o - \frac{z_o d^2\cot\varphi}{4b^2 - d^2\cot^2\varphi}\left(1 - \frac{x_o}{z_o}\cot\varphi - \frac{b}{z_i} - \frac{2br_o}{z_o d}\right) \\ y_r &= y_o \end{cases} \tag{5.7}$$

Thus the error e is

$$e = -\frac{z_o d^2\cot\varphi}{4b^2 - d^2\cot^2\varphi}\left(1 + \cot\varphi\frac{x_o}{z_0} - \frac{b}{z_i} - \frac{2br_o}{z_o d}\right) \tag{5.8}$$

where $-z_o$ is the object distance since $z_o < 0$; z_i is the focus distance; d is the lens diameter; φ is the angle between the z-axis and the board (cf. Fig. 5.7), and usually lies in $[45°, 135°]$ for calibration; p_b intersects the z-axis at $(0, 0, b)$; x_o measures the distance between $\mathbf{s}_o$'s center and the principle point in the projector

plane along the x-axis.

We can see from (5.8) that with a pinhole projector model (i.e. $d = 0$), the error will be 0. This corresponds to our intuitive knowledge. Moreover, there is no defocusing error when the board is vertical (i.e. $\varphi = 90°$). Because in this case, the defocused spot is enlarged uniformly so that E_o and E_r coincide.

In our experiment, the projector's width $W = 1920$ px, its height $H = 1080$ px, the effective focal length (i.e. object distance) $f = -z_o \approx 2000$px, $b = 50$cm, $r_o = 6$px. The maximum error e_{max} is obtained when $v \to \infty$, $\varphi = \varphi_{max} = 135°$ and $x_o = \sqrt{W^2/4 + H^2}$. It can be approximated by:

$$e_{max} \approx \frac{0.52 f d^2}{b^2} \approx 0.29px \tag{5.9}$$

Note that (5.9) can also be used to estimate the smallest admissible board distance for different projectors $b_{min} = d\sqrt{\frac{f}{2e_{max}}}$. When the lens diameter d of a projector is large, the board should be placed further from the projector. This indicates the drawback of our method for projectors with lenses of large sizes.

5.4 Calibration results

Our experimental system consists of a LogiTech C270 webcam (working resolution 640×480), an Epson EB-1771W projector (resolution 1920×1080, lens diameter 8.35mm), a laptop (Intel i7-4510U@2.00GHz CPU, 8GB of RAM) and calibration boards. We use our method to calibrate the system with the projector focusing at various distances, and compare the results with [Audet 2009]. This method is chosen as a reference since it presents competitive results compared to other ones, the source code is available online and it is easy to use.

[Audet 2009] proposes to use a B4 paper (250×353mm) attached to a rigid board as calibration pattern. In order to ease comparison, we use a calibration board of the same size with $2N = 200$ points. We found that with this size, using a 2mm radius for points with 16mm minimum inter-point distance works well. The radius of projected points is $r_o = 6$px.

For the projector we use, a B4 board covers almost all the projection view only at about 50cm from the projector. For other focus distances, three different sizes of pattern (A2, A1, $2 \times A0$) are printed. Marker centers are used in [Audet 2009] as they report to have a better result. For our method, we only use the B4 calibration board to demonstrate that it is much easier to use (cf. Fig. 5.8). At each focus distance, we calibrate 5 times for each method and average the result. For each

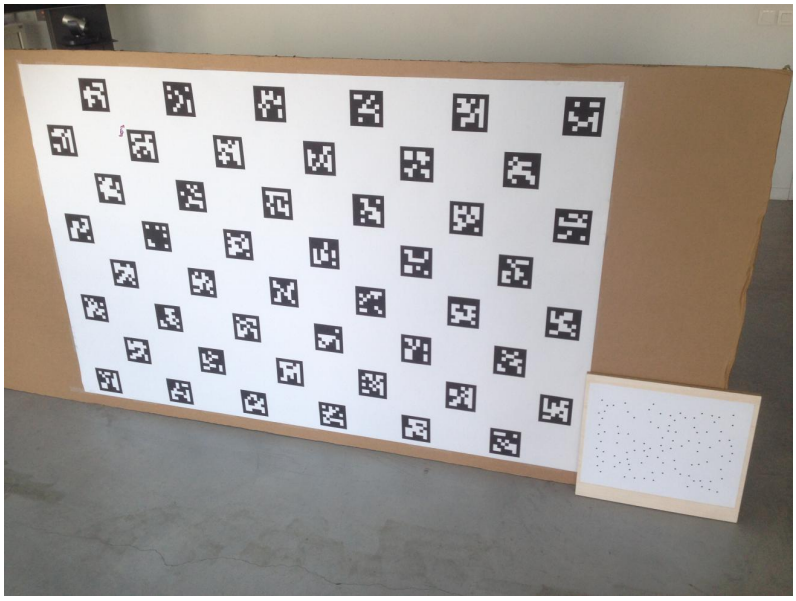


Figure 5.8: Comparison of a $2 \times A0$ calibration board used by [Audet 2009] and a $B4$ calibration board used by our method (bottom-right) for large focus distances ($\geq$ 250cm).

calibration, $K = 10$ images are captured, following [Audet 2009]. Both methods call OpenCV's *calibrateCamera* method for final calibration, with the same set of options and parameters. To prevent the projector's optical properties from changing with the temperature, the projector is pre-warmed 30 minutes before the experiments.

We first show the results of calibration reprojection root mean square errors (RMSE) in Fig. 5.9. RMSE is the most commonly used error measurement for camera and projector calibrations. Compared to [Audet 2009], our average RMSE is smaller but our maximum RMSE is larger. This is reasonable since the more correspondences a method uses, the more likely extreme values appear.

Both methods rely on Zhang's method [Zhang 2000] to find the projector's intrinsic matrix and distortions coefficients by minimizing the sum of reprojection errors of all correspondences. However, as pointed out by [González 2005, Shih 1996], minimizing reprojection errors on calibration data cannot guarantee that the estimated internal parameters are the best ones. Ideally, when no modification is applied to the ProCam system, separated consecutive calibrations should lead to repeatability, i.e. stable intrinsic matrix and lens distortions for both devices.

Fig. 5.10 shows the results on focal lengths estimation at different focus distances. Focal lengths estimated by our method are very stable, with $t(5) = 2.69$ for standard deviation (SD) of f_{px} and $t(5) = 2.49$ for SD of f_{py}, $p < 0.05$. One can clearly observe

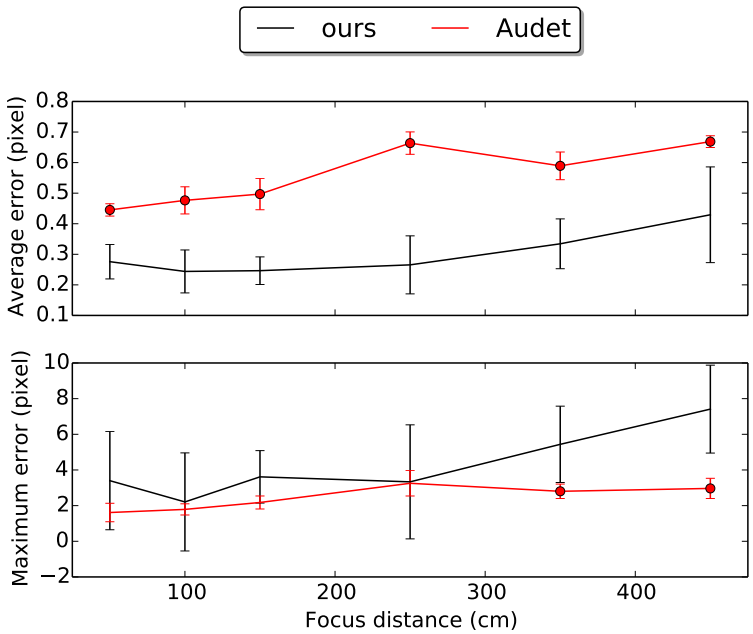


Figure 5.9: Average and maximum reprojection errors (RMSE): A red point on the curve indicates a significant difference between Audet's method and ours, at $p < 0.05$ using a Student's t-test.

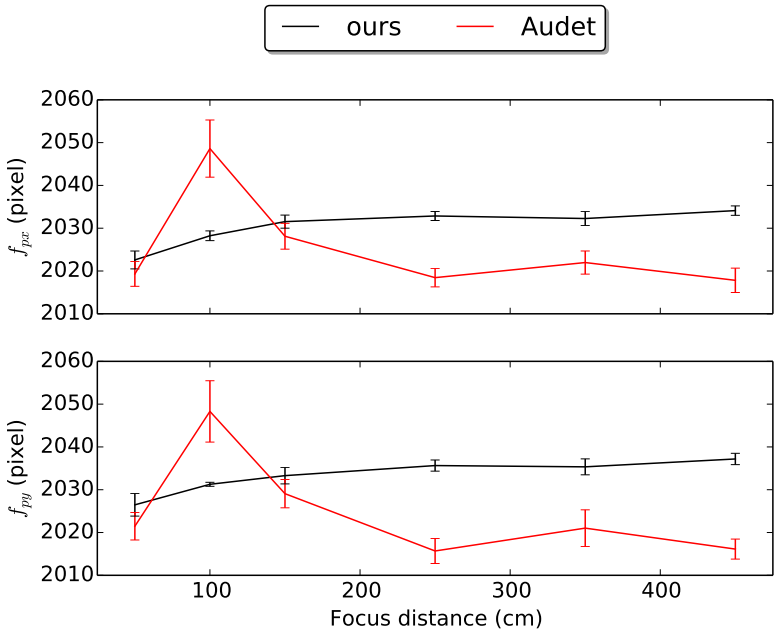


Figure 5.10: Focal length results: our method gives a significantly more stable estimation at $p < 0.05$ using a Student's t-test. It shows the trend of focal length variation.

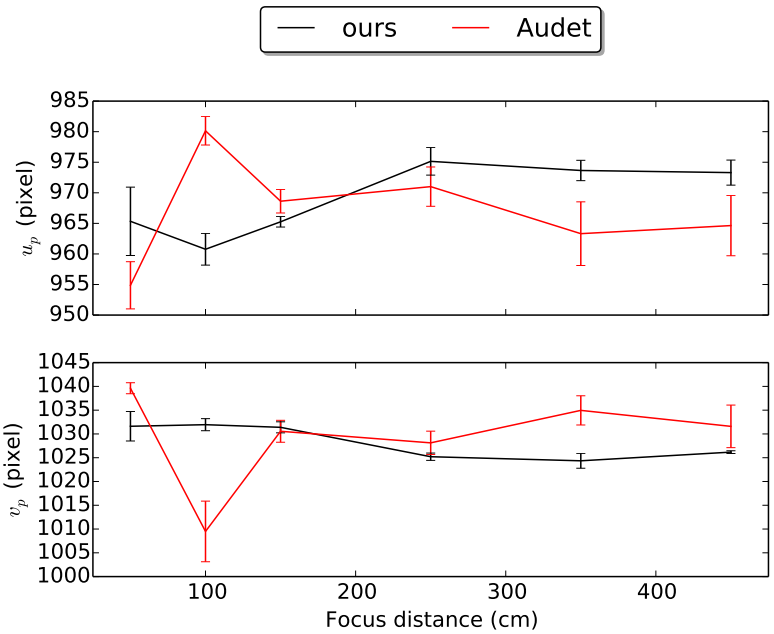


Figure 5.11: Principal point position results: there is no significant difference between our method and Audet's one, despite the size difference between the calibration patterns.

the effective focal length varies according to focus change. In Fig. 5.11, our principal point positions seem more stable as well, but the difference is not significant in t-test ($t(5) = 1.35$ for SD of u and $t(5) = 1.92$ for SD of v, $p > 0.05$). For the camera, our method gives stable results $f_c = 812.4 \pm 0.9$ against Audet's $f_c = 808.1 \pm 4.2$, (f_{cx}, f_{cy} are averaged together).

To make a short summary, our method gives smaller RMSE on calibration data, and more stable intrinsic estimates, despite using a 22.7 times smaller calibration board.

5.5 Augmentation evaluation

In the SAR community, people do not care so much about the true value of intrinsic matrices, nor about the RMSE of calibration data, since real augmentation will hardly lie on these calibration points. They care more about how precise some information can be projected in the focus zone of the projector. In this section, we use calibration-independent data to evaluate this effect.

When relying on [Audet 2009] to augment information at large distances, the user may choose to calibrate either at a short focus distance with a small board (B4

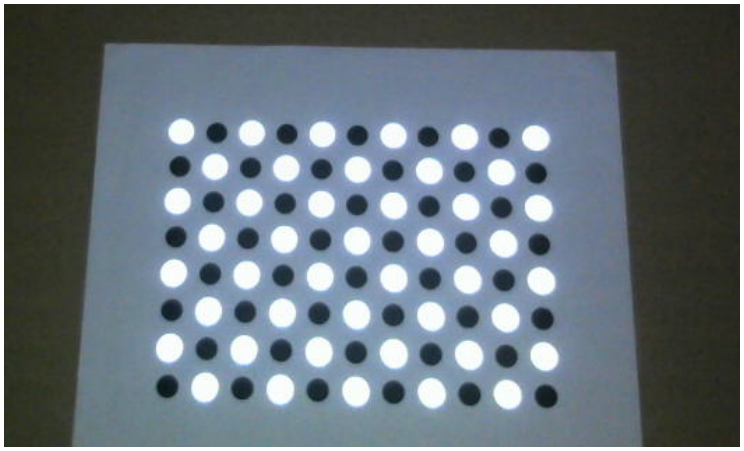


Figure 5.12: Evaluation pattern: (printed) black points are used for localization, (projected) white points are used to measure projection errors.

size) with a loss of precision, or to calibrate at a larger distance with more burden (due to the use of potentially huge calibration boards). As mentioned before, our method offers the advantage to be able to calibrate the ProCam system with only a B4 size calibration board, whatever the focus distance of the projector. In order to draw a fair comparison between our method and Audet's one, three different results are compared here: our method and Audet's calibrating at Correct Focus Distances with different size of boards (i.e. focus distances of the projector during calibration and evaluation are the same, denoted in the figures as Audet-CFD), and Audet's method calibrating with a focus set at 50cm with a B4 board (denoted as Audet-50cm).

We use two asymmetric circle patterns (cf. Fig. 5.12) to investigate projection errors in the focus zone of the projector. Q_b contains the coordinates of all the printed black points while Q_w contains the coordinates of the projected white ones. According to the distance between the ProCam system and the calibration board, different sizes of these patterns are generated. Q_b are printed at exact positions and with known sizes so that we know their true physical positions on the paper. Q_w contains the ground-truth positions at which white points should be projected. They are drawn in an image $I(Q_p)$ to be warped and projected later. After projection, the real positions of the projected circles are measured by the camera and compared to their ground truth values to compute projection errors. A detailed version of this evaluation algorithm is presented in Alg. 10. Its inputs are intrinsic parameters of the camera and the projector, their relative position, as well as two evaluation patterns Q_b and Q_w. Its output is a list of reprojection errors $\{\vec{e_r}\}$ of Q_w.

Algorithm 10 Evaluation

Input: K_c, D_c, K_p, D_p, R, t to be evaluated, Q_b and Q_w
Output: List of reprojection errors $\{\vec{e_r}\}$
$I \leftarrow undistortCameraView(K_c, D_c)$
$Q'_b \leftarrow detectBlackDots(I)*$
$H_{cb} \leftarrow computeHomography(Q'_b, Q_b)$ with $Q'_b = H_{cb}(Q_b)$
$R_{cb}, t_{cb} \leftarrow computePatternPosition(K_c, H_{cb})$
$R_{pb}, t_{pb} \leftarrow coordinateTransformation(R, t, R_{cb}, t_{cb})$
$H_{pb} \leftarrow constructHomography(R_{pb}, t_{pb}, K_p)$
$I_{dp} \leftarrow projectDistortedImage(I(Q_p), D_p, H_{pb})$
$Q'_w \leftarrow detectWhiteDots(I_{dp})*$
$\{\vec{e_r}\} \leftarrow H_{cb}^{-1}(q'_w) - q_w$, for all $(q'_w, q_w) \in (Q'_w, Q_w)$
(*)Perspective and lens' distortions are removed from images. This allows us to find correct circle centers.

Before showing the results, we need to address an issue: how precise can this evaluation algorithm be? The precision depends on the intrinsics of both the camera and the projector . Let us take an example with the camera (the projector would follow the same reasoning): assuming a board parallel to the camera's image plane is positioned at a distance z from the camera's origin (cf. Fig. 5.13). A segment of length L is measured by the camera as being l pixels long, so we have $l = f_c z^{-1} L$. Furthermore, if the angle between the board and the camera's image plane is φ, $l = f_c z^{-1} L \cos(\varphi)$.

We thus see that the smaller the focal length, the smaller l. Assuming $l = 0.5$ pixel (used for illustrative purpose as an approximation of the MSER detector precision), the minimum *on board* difference we can measure is:

$$d_{min} = \frac{z}{2min(f_c, f_p)\cos\varphi} \tag{5.10}$$

RMSE are calculated from the list of reprojection errors $\{\vec{e_r}\}$ in each evaluation image. Considering two series of RMSE $\mathbf{e}_1$ and $\mathbf{e}_2$, if the difference of the average value $\sum \mathbf{e}_1/n_1 - \sum \mathbf{e}_2/n_2$ is less than d_{min}, we consider it to be numerical noise. This effect is represented in Fig. 5.14, 5.17, 5.18 as a gray region encompassing our curve, which means that other RMSE inside this region shows no difference from our result due to the measurement limit. Otherwise, an independent significant test (t-test with different variations) is performed against the result of our method:

$$H_0 : \sum \mathbf{e}_1/n_1 = \sum \mathbf{e}_2/n_2 \tag{5.11}$$

We choose $p = 0.05$, which means the probability that H_0 is wrongly rejected

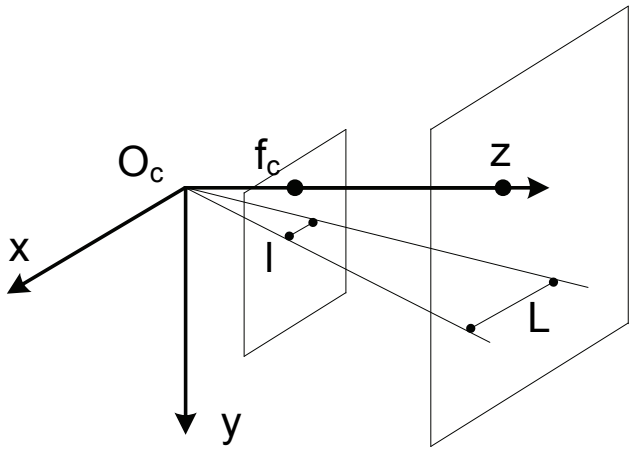


Figure 5.13: Projection of a segment: O_c is the camera's origin. A world segment L is on a board parallel to the camera's image plane at z, its image l is on the image plane. We have $l/L = f_c/z$.

is less than 5%. In our system, $f_c < f_p$, so we have $min(f_c, f_p) = f_c \approx 810$ (cf. Section 5.4).

5.5.1 Focus distance

We first evaluate RMSE at different focus distances (cf. Fig. 5.14). The circle pattern is placed at the center of the focus zone to have the least lens distortions. Although only calibrated at 50cm, Audet-50cm works well for short focus distances ($<$ 250cm), but the RMSE grow rapidly for large focus distances. Except for 250cm, there is no significant performance difference between our method and Audet-CFD although we have to remind the results of Audet-CFD are obtained with huge cumbersome calibration boards (i.e. 1189×1682mm). For our method, the RMSE for a 450cm focus/projection distance are only about 4mm. To have a clear visual difference, we choose 450cm to show the reprojection difference (cf. Fig. 5.15).

5.5.2 Error distribution

Fig. 5.16 shows error distribution in a front view at 250cm. The circle board is placed at four different places to cover the whole projector view. Both our method and Audet-CFD have small errors while Audet-50cm gives large errors especially for small y values. We can notice that Audet-CFD generally has smaller errors than ours in Fig. 5.16 while our method shows smaller RMSE at 250cm in Fig. 5.14.

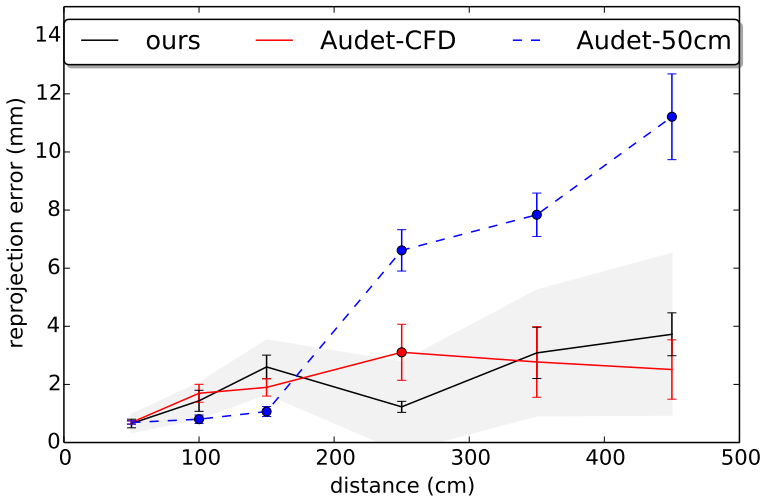


Figure 5.14: RMSE with different focus distances. Color dots mean that RMSE significantly differ from our result: H_0 of (5.11) is rejected.

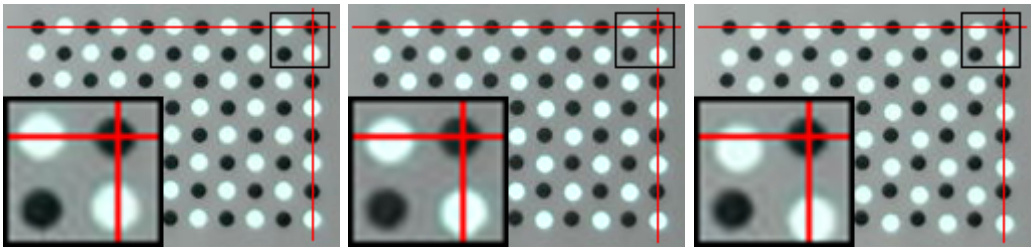


Figure 5.15: Reprojection error in rectified views (focusing at 450cm). From left to right: ours, Audet-CFD, Audet-50cm. Red lines are drawn to show points alignment. Circles' diameter is 48mm.

However, this is not contradictory, since the error difference reaches the limits of the measurement system.

5.5.3 Perspective and depth

At last, we study the influence of perspective angle and depth in the projector's focus zone. The projector is always focusing at 250cm. Fig. 5.17 shows that all methods perform worse under large oblique angles than under small ones. Audet-50cm is significantly worse in many cases while our method and Audet-CFD give almost the same result, yet again at the expense of the size of the calibration board. Fig. 5.18 shows that when further away from the projector, Audet-50cm has a poor performance.

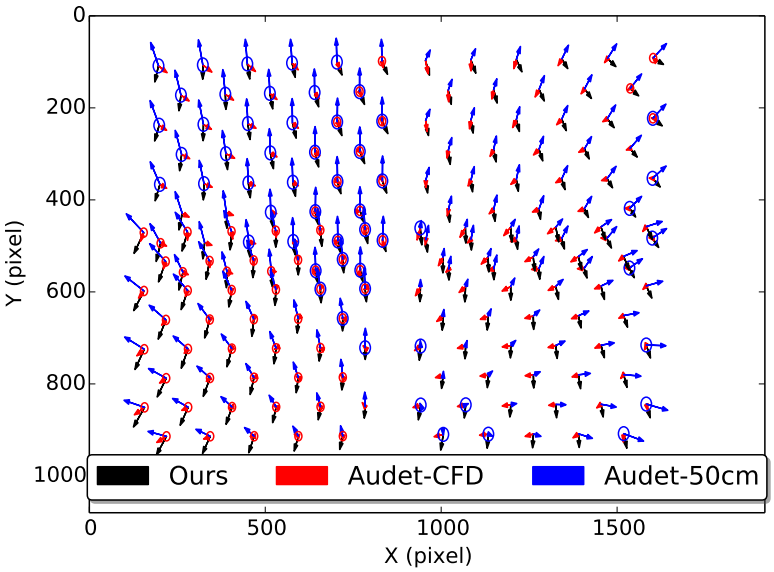


Figure 5.16: Error distribution in projector's view (focusing at 250cm). Reprojection errors (instead of RMSE) are used in the t-test: Color circles indicate significant differences from our result.

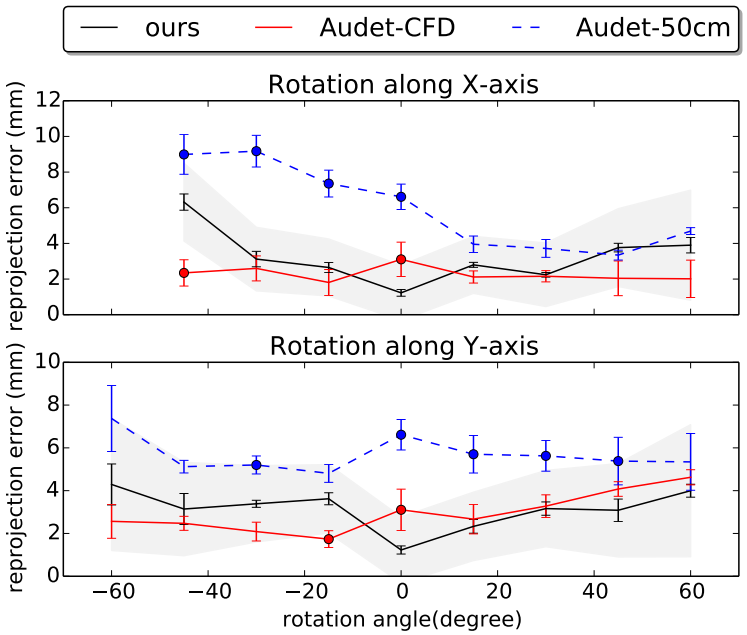


Figure 5.17: RMSE with various rotations (focusing at 250cm). Color dots mean RMSE significantly differ from our result: H_0 of (5.11) is rejected.

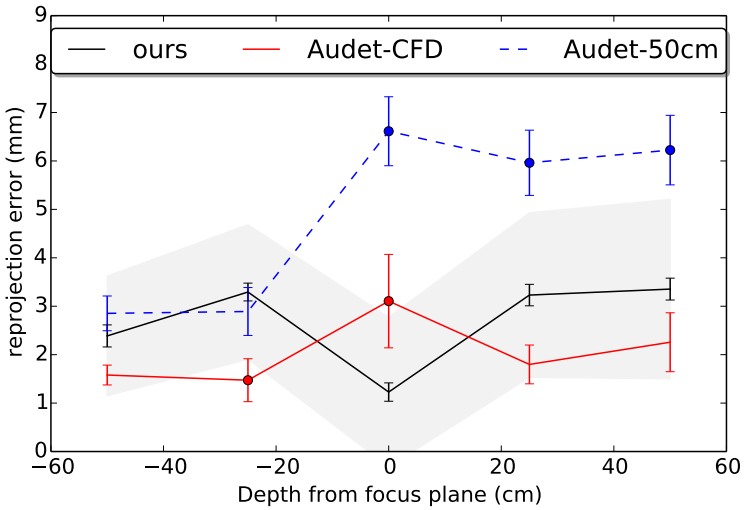


Figure 5.18: RMSE with various depths (focusing at 250cm). Color dots mean RMSE significantly differ from our result: H_0 of (5.11) is rejected.

5.6 Conclusion

This chapter presented a practical method to calibrate a projector-camera system, by using the LGC method introduced in Chapter 4. The method can calibrate a ProCam system despite the focus distance of the projector. It is user-friendly and interactive in real-time. We evaluated the calibration results with calibration-independent data. The reprojection error of the system is competitive with regard to the state-of-art, but the method only demands a small calibration board and simple manipulations even at large focus distances (more than 2.5m). Results also show that the calibrated intrinsic matrix of the projector is more stable than other state-of-the-art methods [Audet 2009]. The drawback of our method is that it potentially has larger system error for projector with large lenses according to (5.9). Nevertheless this can be solved by calibrating the system at a slightly larger distance. Since defocus projections are tolerated by our method, the calibration board would still be smaller than state-of-the-art methods [Audet 2009].

Chapter 6

Surface of revolution reconstruction from 3D data

Contents

This chapter presents the work realized during a three-month internship in Kyushu University in Japan, and has been presented at *International Conference on 3D Vision (3DV)* in 2016. It tackles scene understanding from 3D data, which is completely different from the previous chapters. Scene understanding from 3D data is an important research topic in computer vision and robotic perception because it can be applied to various types of systems such as robotic grasping, model simplification, augmented reality, etc. Existing methods on this topic can be classified into two main families: top-down and bottom-up approaches [Nguyen 2015]. Top-down approaches are often referred to as object detection, since they aim at

detecting pre-known models. Bottom-up approaches are sometimes called geometric reconstruction methods. They detect primitive shapes in a scene and establish relationships between these shapes whenever possible.

Primitive detection and localization is a crucial step in bottom-up approaches: it represents the first step of those methods and determines their application range. However, existing approaches can only deal with a limited type of primitive shapes such as: planes only [Nguyen 2015], cylinders only [Qiu 2014] or thin-structures [Song 2015]. More complex primitives such as spheres, cones or tori [Schnabel 2007] can be detected but those techniques still remain limited to pre-defined parametric shapes. It is important for applications in reverse engineering or in the construction field to deal with various types of geometric primitive shapes. Also, on-line scene understanding is desirable as dense SLAM (i.e. SLAM from RGB-D data) runs in real-time from 3D data [Tateno 2016].

In this chapter, we aim at detecting and reconstructing surfaces of revolution (SoR) in a cluttered scene in real-time. SoR can represent the majority of primitives and are quite common in man-made objects due to their ease of production. Since the most crucial step in detecting a SoR is to estimate its rotation axis, we propose a fast and accurate method, which boils down the estimation of the rotation axis to a one dimension search. The main contributions of this chapter are summarized as follows:

- a fast and accurate estimation of SoR axes from 3D data
- a framework for detecting, localizing and reconstructing SoR in a cluttered scene in real-time.

6.1 Related work

For semantic 3D understanding, structural modeling using geometric primitives has been investigated in the literature. Qiu et al. [Qiu 2014] use parallel cylinder detection to reconstruct an industrial pipeline system. Song et al. [Song 2015] use beams and planes to reconstruct thin structural systems, such as chairs. Thanh et al. [Nguyen 2015] use planes to find semantic structures in a scene. In order to model the structure of more complex objects, Schnabel et al. [Schnabel 2007] propose a RANSAC-like method which can detect cones, cylinders, spheres and tori to reconstruct more types of primitive shapes. Taguchi et al. [Taguchi 2015] improved this method by introducing distance field functions for efficient computation. Drost et al. [Drost 2015] propose to use Hough Transform to detect cylinders and spheres.

All these methods can detect a limited set of parametric shapes, and are not designed to work in real-time.

A surface of revolution (SoR), or rotational object, is formed by rotating a 2D curve (i.e. the profile) in 3D space with respect to a 3D line (i.e. the rotation axis), thus it is axis-symmetric and is a more general representation of most geometric primitives. Once the position of its rotation axis is determined, the profile can be easily calculated. Existing approaches are dedicated to the estimation of the rotation axis of a SoR, and can be classified as direct, iterative and brute force methods. For direct methods, Pottmann et al. [Pottmann 1999] use 3D line geometry to classify surfaces and find the rotation axis of a SoR by solving a 6D eigenvalue problem. For iterative methods, [Lou 2009],[Willis 2003] and [Pavlakos 2015] use ICP-like method [Besl 1992a]. They first estimate the profile function of a SoR by using a presumed rotation axis, thus creating an initial model of the SoR. Then, they try to align this model with the point cloud, which results in a better estimation of the axis for the next iteration. Brute force methods use the fact that the intersection between the SoR and a plane perpendicular to its rotation axis is a circle. Han et al. [Han 2012] use planes containing the normal of a surface point to cut a SoR and evaluate the intersecting curve. The plane which gives the least curvature variation is used to calculate the rotation axis. Mara et al. [Mara 2006] use a set of parallel planes to cut a SoR. Then circles are fitted on each plane and the variation of circle centers across all planes is calculated. This set of planes is then rotated in a 2-dimensional search space so that the set which gives the least variation of circle centers is used to calculate the rotation axis.

Although direct methods are quite fast, they cannot find accurate results if input data contains even moderate noise (cf. Section 6.5). Iterative methods use the whole point cloud, fit the SoR profile at each step, and are thus very time consuming. Brute force methods are time consuming as well.

To reconstruct SoR in real-time, we propose a new iterative method to improve the efficiency while still ensuring a good accuracy. It is inspired by the method of Mara et al. [Mara 2006] that uses parallel planes to cut a SoR. However, while [Mara 2006] uses a 2-dimensional brute force search, our method finds the axis by solving a 1-dimensional objective function. The same cutting operation (cf. Section 6.3.2) is repeated almost a hundred times in [Mara 2006] to reach a solution, but is only required at most 5 times in our method.

6.2 Surface of revolution axis estimation

The procedure of reconstructing a SoR in our method is summarized as follows:

1. Estimate an initial rotation axis l_0 using Pottmann's method [Pottmann 1999] (denoted as *P-method* afterwards)
2. Improve the rotation axis estimation (cf. Alg. 11)
   (a) Use parallel planes to cut the SoR along l_0
   (b) For each plane: find the center of the intersecting curve (cf. Fig. 6.1)
   (c) Fit a line to all centers
   (d) Rotate the cutting planes and goto step 2a if the line is not perpendicular to the planes
3. Reconstruct the SoR by finding its profile (cf. Section 6.3.1)

A fast and accurate iterative step 2 is our main contribution. Usually, the direction of a rotation axis l can be determined by two Euler angles and many existing methods use two dimensional search, such as Mara et al. [Mara 2006]. We show that the search space can be reduced to one dimension, and that the objective function is easy to solve. Both of them make the algorithm more efficient. Steps 1 and 3 can be seen as pre-process and post-process respectively.

In this section, we first explain the idea of step 2 and then mathematically prove it. The implementation of step 2 and step 3 are then presented in Section 6.3.

6.2.1 Basic idea

We first show that, given an initial guess l_0 obtained from the P-method, the search space of the direction of the rotation axis is unidimensional. In Fig. 6.1, we use a cone $\mathbf{R}$ to represent a general SoR for the sake of simplicity. The solid line l_g is the ground truth of its rotation axis. Line l_0 represents the initial guess. A plane P_s containing l_g can be defined by the normal $\vec{l_g} \times \vec{l_0}$, where the unit vector $\vec{\cdot}$ represents the line's direction. By rotating $\vec{l_0}$ inside P_s, it is possible to find $\vec{l_g}$. Therefore, the search space is reduced to one dimension. Note that we use a 2D sketch of P_s in Fig. 6.2 to represent Fig. 6.1 because all useful information is contained in P_s.

We now show that P_s can be determined without knowing l_g. We first create n parallel planes P_j ($j = 1..n$) with normal $\vec{l_0}$ to cut $\mathbf{R}$, which results in a set of 2D curves C_j. Since C_j are symmetric curves with respect to P_s, their centers O_j should lie on P_s. We then project the centers of all C_j on an arbitrary plane, e.g.

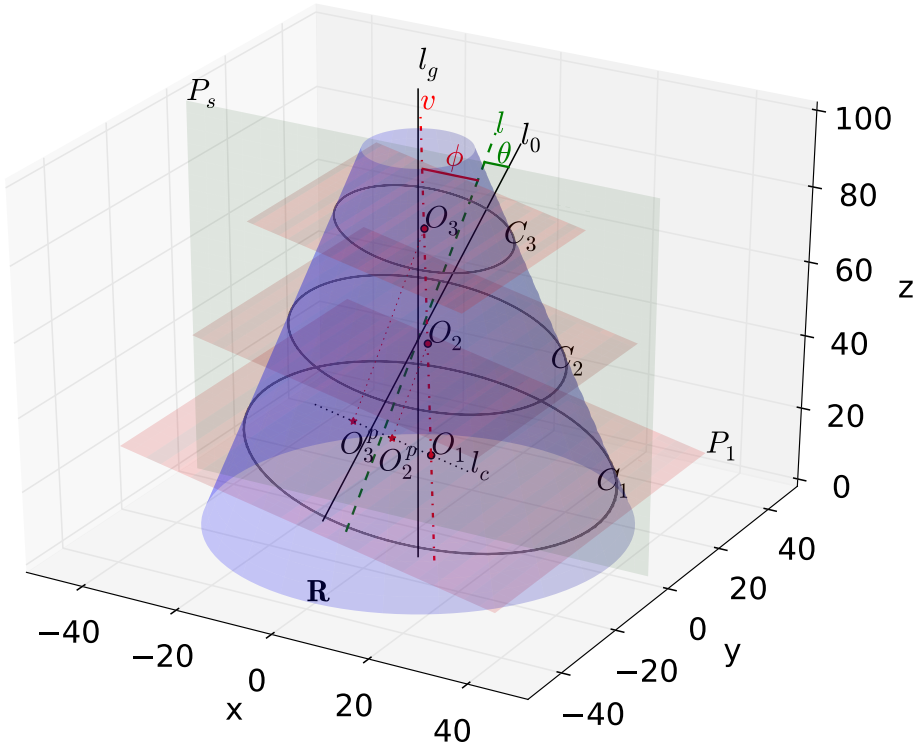


Figure 6.1: Symmetry of SoR and planes

P_1. These projections O_j^p ($O_1^p = O_1$) form a line l_c (cf. Fig. 6.1), which is also the intersection between P_s and P_1. By finding $\vec{l_c}$, one can determine P_s.

Next, we derive the objective function of our method. Let us assume that $\vec{l}$ is rotated from $\vec{l_0}$ inside P_s by angle θ (cf. Fig. 6.1-6.2). The set of parallel planes P_j is also rotated so that their normal is $\vec{l}$. A line v is fitted to all points O_j. Intuitively, if $\vec{l} = \vec{l_g}$, all C_j should be circles because of symmetry, and their centers O_j should lie on l_g. It means that $\vec{l} = \vec{l_g} \Rightarrow \vec{v} = \vec{l_g} = \vec{l}$. In the next section, we show that the converse is true if $\mathbf{R}$ is not a sphere. That is to say $\vec{v} = \vec{l} \Leftrightarrow \vec{l} = \vec{l_g}$ for non-spheres. In Fig. 6.2, unit vectors inside P_s can be represented by 2D vectors. By abusing notations, we still use $\vec{\cdot}$ to refer to these 2D vectors, so their cross products are real numbers. Finally, the objective function is:

$$\begin{aligned} \phi(\theta) &= \sin^{-1}(\vec{l} \times \vec{v}) \\ \text{with } \theta &= \sin^{-1}(\vec{l_0} \times \vec{l}) \end{aligned} \tag{6.1}$$

Both $\phi(\theta)$ and θ are signed angles. Our objective is to find θ such that $\phi(\theta) = 0$.

Once $\vec{l_g}$ is found, we can use the average of O_j as a point on the axis. This gives

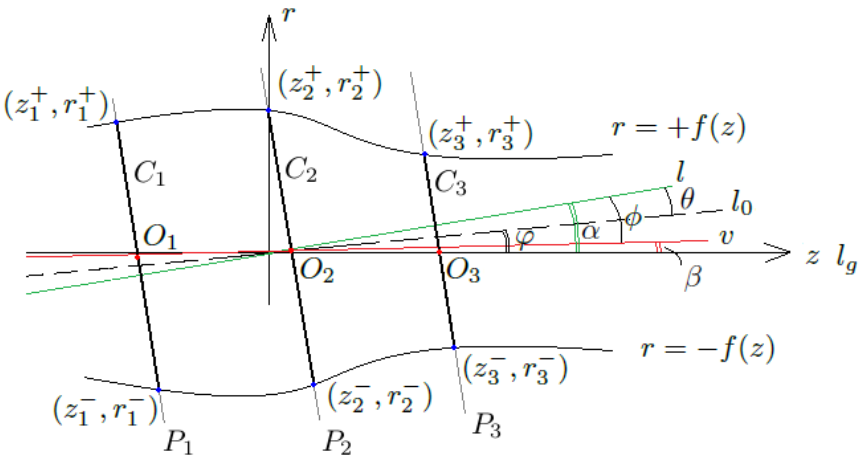


Figure 6.2: Sketch of P_s: All useful quantities are inside P_s

us the final rotation axis as:

$$l_g = (\vec{l_g}, \frac{1}{n}\sum_{j=1}^{n} O_j) \tag{6.2}$$

6.2.2 Approximately linear objective function

In this section, we prove that the objective function ϕ (6.1) is approximately linear with respect to θ. This is the core of our method since it guarantees its fast convergence.

The unknown l_g is chosen to be the z-axis in Fig. 6.2 for an easier deduction. The profile of the SoR in this plane can be expressed as $r = \pm f(z)$. Parallel planes intersect with the z-axis at z_j. The 2D projection of C_js on P_s can be represented by segments between (z_j^+, r_j^+) and (z_j^-, r_j^-). φ is defined as the offset angle of initial guess $\vec{l_0}$ from $\vec{l_g}$:

$$\varphi = \sin^{-1}(\vec{l_g} \times \vec{l_0}) \tag{6.3}$$

Since θ is the angle from $\vec{l_0}$ to $\vec{l}$, so when $\theta = -\varphi$, we have $\vec{l} = \vec{l_g}$ and thus the objective function $\phi(\theta) = \phi(-\varphi) = 0$. To derive the formula of $\phi(\theta)$, two intermediate angles are used:

$$\begin{aligned} \alpha &= \sin^{-1}(\vec{l_g} \times \vec{l}) \\ \beta &= \sin^{-1}(\vec{l_g} \times \vec{v}) \end{aligned} \tag{6.4}$$

It is easy to derive that:

$$\begin{aligned} \theta &= \alpha - \varphi \\ \phi &= \beta - \alpha \end{aligned} \tag{6.5}$$

By assuming that α is small, the profile near z_j can be approximated by its first order Taylor expansion:

$$r = \pm f(z) \approx \pm \big(f(z_j) + f'(z_j)(z - z_j)\big) \tag{6.6}$$

The equation of plane P_j can be expressed as:

$$r = \tan(\frac{\pi}{2} + \alpha)(z - z_j) \tag{6.7}$$

The intersection points (z_j^+, r_j^+) can be calculated by solving (6.7) and using $r = +f(z)$ from (6.6). Similarly, (z_j^-, r_j^-) can be calculated from (6.7) and using $r = -f(z)$ from (6.6). The center point $O_j(z_j^*, r_j^*)$ lies in the middle of the $[(z_j^-, r_j^-)$, $(z_j^+, r_j^+)]$ line segment and thus can be derived as:

$$\begin{aligned} z_j^* &= \frac{1}{2}(z_j^- + z_j^+) = z_j + \frac{f(z_j)f'(z_j)}{\cot^2\alpha - f'^2(z_j)} \\ r_j^* &= \frac{1}{2}(r_j^- + r_j^+) = -\frac{\cot\alpha f(z_j)f'(z_j)}{\cot^2\alpha - f'^2(z_j)} \end{aligned} \tag{6.8}$$

Since v is an interpolation of all O_j, its slope k in the plane can be expressed as follows:

$$k = \tan\beta = -\frac{\sum\limits_1^n (z_j^*)^2 - n\left(\sum\limits_1^n \frac{z_j^*}{n}\right)^2}{\sum\limits_1^n z_j^* r_j^* - n\left(\sum\limits_1^n \frac{z_j^*}{n}\right)\left(\sum\limits_1^n \frac{r_j^*}{n}\right)} \tan\alpha \tag{6.9}$$

Since α is small, the equation above can be simplified by the Taylor expansion and by using the approximation $\tan\alpha \approx \alpha$ along with (6.8):

$$k \approx -\frac{n\sum\limits_1^n z_j f(z_j)f'(z_j) - \left(\sum\limits_1^n z_j\right)\left(\sum\limits_1^n f(z_j)f'(z_j)\right)}{n\sum\limits_1^n z_j^2 - \left(\sum\limits_1^n z_j\right)^2}\alpha \tag{6.10}$$

When β is small, we can approximate that $k = \tan\beta \approx \beta$. Let $\xi_j = z_j - \frac{1}{n}\sum\limits_1^n z_j$, we

have $\sum_{1}^{n} \xi_j = 0$. As a consequence, (6.10) can be simplified as:

$$\beta \approx -\frac{\sum_{1}^{n} \xi_j f(\xi_j + \bar{z}) f'(\xi_j + \bar{z})}{\sum_{1}^{n} \xi_j^2} \alpha = -(A-1)\alpha \tag{6.11}$$

where

$$A = \frac{\sum_{1}^{n} \xi_j f(\xi_j + \bar{z}) f'(\xi_j + \bar{z})}{\sum_{1}^{n} \xi_j^2} + 1 \tag{6.12}$$

Thus, by using (6.5) and (6.11), the objective function $\phi(\theta)$ can be written as:

$$\phi(\theta) = \beta - \alpha = -A\alpha = -A(\theta + \varphi) \tag{6.13}$$

When $\vec{l}$ is rotated to the ground-truth $\vec{l}_g$, then $\theta = -\varphi$ and thus $\phi(\theta) = 0$, which corresponds to our intuitive knowledge. In (6.13), φ is an unknown variable depending on l_0. Although A depends on ξ_j (cf. (6.12)), by letting $\Delta\xi = \frac{\xi_n - \xi_1}{n-1}$, we can show that when $n \to \infty$:

$$\begin{aligned} A &= \frac{\sum_{1}^{n} \xi_j f(\xi_j + \bar{z}) f'(\xi_j + \bar{z}) \Delta\xi}{\sum_{1}^{n} \xi_j^2 \Delta\xi} + 1 \\ &\approx \frac{\int_{\xi_{min}}^{\xi_{max}} \xi_j f(\xi_j + \bar{z}) f'(\xi_j + \bar{z}) d\xi}{\int_{\xi_{min}}^{\xi_{max}} \xi_j^2 d\xi} + 1 \end{aligned} \tag{6.14}$$

It shows that A is a discrete approximation of an integral. Thus although A depends on ξ_j, it does not vary much when n is big. In other words, the objective function $\phi(\theta)$ is approximately linear.

We now show that the solution $\phi(\theta) = 0$ is unique except for spheres and that it corresponds to the rotation axis. From (6.13), we can see that $A = 0 \Rightarrow \phi(\theta) \equiv 0$. By solving $A = 0$ from (6.14), we find the profile function is $f(z) = \pm\sqrt{(C - z^2)}$, where C is an arbitrary constant. This equation represents a sphere of radius $\sqrt{C}$. $\phi \equiv 0$ means a sphere can have its rotation axis in any direction, which is obvious. But for other surfaces, since $A \neq 0$, the solution of $\phi(\theta) = 0$ is uniquely $\theta = -\varphi$ by (6.13). It means that by finding the unique solution of $\phi(\theta) = 0$, one can find its

unique rotation axis.

6.3 Implementation details

3D data of real objects are often obtained by using RGB-D sensors (i.e. Kinect, RealSense). They can have various representations, such as point clouds or Signed Distance Fields (SDF). However, captured data may be noisy. In this section, we explain implementation details by considering noisy data. The outline of the algorithm is first introduced in Section 6.3.1. $findCircleCenters$ is a basic and important operation in the algorithm. Its detail is explained in Section 6.3.2. At last, we solve the objective function numerically in Section 6.3.3.

6.3.1 Algorithm

The outline of rotation axis estimation is shown in Alg. 11. It takes a 3D representation (e.g. a point cloud, a SDF) of a SoR and an initial axis estimation l_0 obtained from [Pottmann 1999] as inputs and returns the rotation axis l_e.

Algorithm 11 Rotation axis estimation

Input: 3D data of SoR, initial rotation axis l_0 estimated using [Pottmann 1999]
Output: SoR rotation axis l_e
Parameter: number of cutting planes n_1, i_{max}
$i = 0, \theta_0 = 0, valuePair = \emptyset$
while $i < i_{max}$ **do**
  if $i \neq 0$ **then**
    $\theta_i = \theta_{i-1} - \phi_{i-1}/A_{i-1}$
    $\vec{l_i} \leftarrow rotate(\vec{l_0}, P_s, \theta_i)$
  $\mathbf{O}, \mathbf{w} \leftarrow findCircleCenters(l_i, n_1)$ or $fail$
  if $i = 0$ **then**
    $P_s, \delta \leftarrow findSymmetricPlane(\mathbf{O})$
  $\vec{v_i} \leftarrow findCenterLineDirection(\mathbf{O}, \mathbf{w})$
  $\phi_i = \sin^{-1}(\vec{l_i} \times \vec{v_i})$
  $valuePair.append(\theta_i, \phi_i)$
  if $|\phi_i| < \varepsilon$ **then**
    return $\vec{l_e} = \vec{l_i}$
  else
    if $i = 0$ **then**
      Estimate A_0 with $(\mathbf{O}, \mathbf{r})$ from (6.20)
    else
      $A_i \leftarrow calculateSlope(valuePair)$
  $i = i + 1$
Find smallest $|\phi_i|$ in $valuePair$ and return $\vec{l_e} = \vec{l_i}$

The process of cutting the SoR by parallel planes and finding the centers of intersecting curves is frequently used in our method. Since the angle between the presumed axis l_i and the ground-truth l_g is usually smaller than 10° (cf. Fig. 6.7), we use circle fitting to find the centers O_j. We call this process *findCircleCenters*.

$$\mathbf{O}, \mathbf{w}, \mathbf{R}, \bar{\mathbf{e}} \leftarrow findCircleCenters(l, n) \tag{6.15}$$

This process takes the presumed axis l and the number of planes n as inputs and returns n_v valid circles as output, with circle centers $\mathbf{O} = \{O_1, O_2, ..., O_{n_v}\}$, circle radii $\mathbf{R} = \{R_1, R_2, ..., R_{n_v}\}$, weights $\mathbf{w} = \{w_1, w_2, ..., w_{n_v}\}$ and average geometric errors $\bar{\mathbf{e}} = \{\bar{e}_1, \bar{e}_2, ..., \bar{e}_{n_v}\}$. n_v is the number of valid circles, which may be smaller than n. A detailed explanation is provided in Section 6.3.2. Different n can be used as input in different cases. In Alg. 11, we use n_1 in order to distinguish n_2 that is used for reconstruction (see later in this Section).

In Alg. 11, *findSymmetricPlane* is the process of finding P_s. It takes all valid circle centers $\mathbf{O}$ as input and gives P_s and δ as outputs. δ represents the incertitude of circle centers detection, which is explained in Section 6.3.3. *findCenterLineDirection* is the process of finding the direction of v (cf. Fig. 6.2). It takes circle centers $\mathbf{O}$ and their weights $\mathbf{w}$ as inputs. It fits a line to points $\mathbf{O}$ by using different weights $\mathbf{w}$ and returns $\vec{v}$ as output. *calculateSlope* is the process of finding the slope A of $\phi(\theta)$. It takes as input the previous $i+1$ value pairs $valuePair = \{(\theta_0, \phi_0), (\theta_1, \phi_1), .., (\theta_i, \phi_i)\}$, fits a line to them and returns the slope A of this fitted line.

After finding $\vec{l_e}$, the whole object can be reconstructed. $findCircleCenters$ is used as well, but with inputs $\vec{l_e}$ and n_2. n_2 can be larger than n_1 in Alg. 11 for a finer reconstruction result. A surface's intrinsic coordinate system is constructed, with origin O_e being the average of O_j and the direction of z-axis $\hat{z}_e = \vec{l_e}$. Within this coordinate system, the profile function $f(z_e)$ is fitted by splines, thus finishing the SoR reconstruction.

The global reconstruction error e, is defined as the average of geometrical error $\bar{\mathbf{e}}$ (cf. Section 6.3.2).

$$e = average(\bar{\mathbf{e}}) \tag{6.16}$$

6.3.2 Plane cutting and circle fitting

We use parallel planes separated by an equal interval in *findCircleCenters*. When point clouds are used as raw data with resolution (i.e. average inter-point distance)

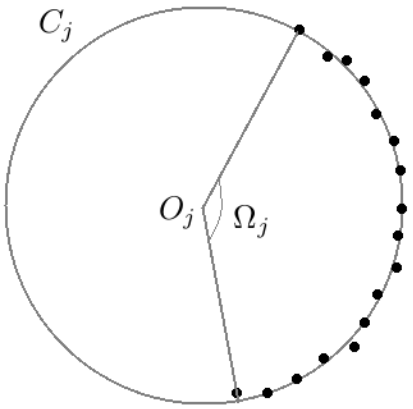


Figure 6.3: Angular span of data points on a fitted circle: black points represent data points in plane P_j, C_j is the fitted circle, Ω_j is the angular span of data points.

τ, a point is considered inside a plane if its distance to the plane is less than 0.5τ. If a SDF representation (with voxel size τ) is used, a ray casting method can be used to find points inside a specified plane. Circles are fitted to points in the same plane. We define the *geometric error* as the distance between a point and the fitted circle in the same plane. A point is considered as an inlier if its geometric error is less than τ.

Since scanned 3D data contain lots of defects, such as incomplete scanning and noise, some planes may contain data with better quality than others. To compensate those limitations, a circle is considered as valid only when its inlier ratio of circle fitting is more than 80%, and each valid circle C_j is associated with a weight w_j according to its fitting result. Additionally, planes near the two ends of a SoR are more easily influenced by inaccurate axis direction estimation, so we only use empirically the middle 70% region of the surface for axis estimation. The weight w_j for a valid circle is calculated by:

$$w_j = \frac{\Omega_j}{\bar{e}_j} \tag{6.17}$$

where Ω_j is the angular span of data points on the fitted circle as shown in Fig. 6.3, $\bar{e}_j$ is the average geometric error of all inliers in the current plane.

If *findCircleCenters* finds less than 2 valid circles, it means that the target surface is not a true SoR or the data is too noisy to process. The algorithm fails in this case.

6.3.3 Determining P_s and solving $\phi(\theta)$

The basic idea of calculating P_s has been explained in Section 6.2.1. Assuming that the incertitude of center detection is δ, the center projections O_j^p in Fig. 6.1 will lie in a thick line of width 2δ. By using Principle Component Analysis (PCA) on these center projections, the major eigenvector can be regarded as a good approximation of $\vec{l_c}$ (cf. Fig. 6.1) while the minor eigenvalue λ_{min} is the sum of squared distances between detected circle centers and P_s. Thus, δ can be approximated by:

$$\delta \approx \sqrt{\lambda_{min}/n_v} \tag{6.18}$$

where n_v is the number of valid circles.

Although $\phi(\theta)$ has a simple form, since the scanned 3D data is noisy and $\phi(\theta)$ is not exactly linear (cf. Section 6.2.2), we still have to rely on an iterative approach to solve $\phi(\theta) = 0$. Recall Newton's method:

$$\theta_{i+1} = \theta_i - \frac{\phi(\theta_i)}{\phi'(\theta_i)} = \theta_i - \frac{\phi(\theta_i)}{A_i} \tag{6.19}$$

For the i-th step, θ_i is known and $\phi(\theta_i)$ is evaluated. Owing to its approximate linearity, $\phi'(\theta_i) = A_i$ can be approximated by fitting previous $i+1$ data by using *calculateSlope*. For the first step $i = 0, \theta_0 = 0$, we solve the primitive function (6.12) to find an approximation of A_0:

$$\begin{aligned} A_0 &\approx 3\frac{f^2(z_{max}) + f^2(z_{min}) - \frac{2}{n_v}\sum_1^{n_v} f^2(z_j)}{(z_{max} - z_{min})^2} + 1 \\ &\approx 3\frac{R_{max}^2 + R_{min}^2 - \frac{2}{n_v}\sum_1^{n_v} R_j^2}{D^2} + 1 \end{aligned} \tag{6.20}$$

where D is the distance between the two most distant planes $(z_{max} - z_{min})$. When α is small, P_j is almost perpendicular to l_g, so we use the the radius of fitted circle R_j to approximate $f(z_j)$ (cf. Fig. 6.2) in the second approximation.

The iterative process can be terminated when $|\phi(\theta)|$ is small enough, e.g. $\leq 10^{-3} rad$. However, since circle center detection cannot be more precise than δ according to (6.18), the estimation of the rotation axis cannot be more precise than $\tan^{-1}(\delta/D)$. Since $\delta \ll D$, δ/D can be used instead of $\tan^{-1}(\delta/D)$. Therefore, the iterative process is terminated when $|\phi(\theta_i)| < \varepsilon = \max(\delta/D, 10^{-3})$. Practically, we found that this method usually converges in 2-3 steps. However, in the special case where a point cloud is used, $\phi(\theta_i)$ may not be continuous and $|\phi(\theta_i)| < \varepsilon$ may

never be achieved. This leads the process to oscillate around $\phi(\theta_i) = 0$. In order to avoid this, we set the maximum iteration number $i_{max} = 5$. If i_{max} is reached but $|\phi(\theta_{i_{max}})| \geq \varepsilon$, the l_i giving the smallest $|\phi|$ will be chosen as l_e.

6.4 Real-time SoR reconstruction

We propose a workflow to reconstruct SoR from dense SLAM data. It uses real-time segmentation result from the work of [Tateno 2015], where surfel-based representation is used. A surfel is a representation of a small surface patch at position $\vec{p}(x, y, z)$ in 3D space, with its normal direction $\vec{n}$, its confidence radius r and one segment label. Surfels having the same segment label make up a segment. Usually, each segment represents a single object in the scene. We should mention that, although we use surfel-based representation in this section, our algorithm can be easily adapted to deal with any other SLAM data structure.

Although this workflow can deal with SoR, planes and spheres, we only focus on SoR to highlight our contribution. A method to distinguish among different types of surfaces is presented in Section 6.4.1. It allows us to find segments which represent SoR. The detailed algorithm to reconstruct SoR using [Tateno 2015] as our SLAM data provider at each frame is presented in Section 6.4.2.

6.4.1 Segment classification

We use a two-step method to classify each segment. A surfel at position $\vec{p}$ having a normal $\vec{n}$ is denoted by $(\vec{p}, \vec{n})$. Let $\vec{r} = (\vec{p} \times \vec{n}, \vec{n})$ be a 6 dimension vector. For all N points belonging to one segment, the following two matrices can be calculated:

$$\begin{aligned} L &= \frac{1}{N} \sum_{j=1}^{N} \vec{n}_j \vec{n}_j^T \\ M &= \frac{1}{N} \sum_{j=1}^{N} \vec{r}_j \vec{r}_j^T \end{aligned} \tag{6.21}$$

where L is a 3×3 matrix and M is a 6×6 matrix. Let $\{\lambda_{L,i} : i = 1, 2, 3\}$ be

f_t	2	1		
f_r	-	> 2	1	0
Type	Plane	Sphere	SoR	Complex

Table 6.1: Different surface types according to f_t and f_r

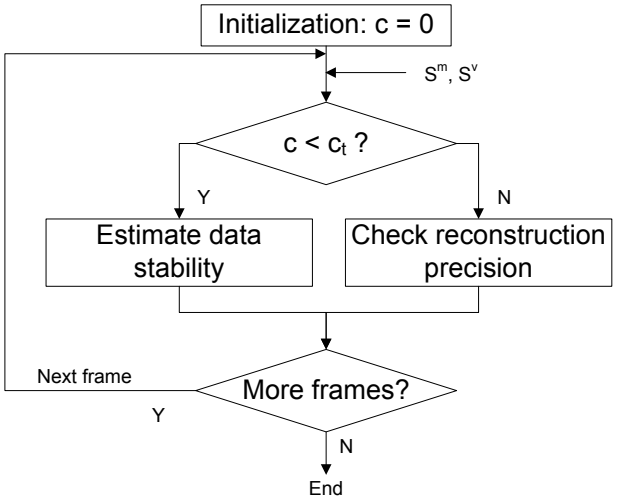


Figure 6.4: Workflow of real-time surface reconstruction for SoR.

the eigenvalues of L and f_t be the number of small eigenvalues in $\{\lambda_{L,i}\}$. According to [Lou 2009], the segment can be translated in f_t directions without changing the distance function values nearby. We can say that the segment has f_t translation free directions. Let $\{\lambda_{M,i} : i = 1, 2, 3\}$ be the eigenvalues of problem $M\vec{x} = \lambda D\vec{x}$, where $D = diag(1, 1, 1, 0, 0, 0)$, and f_r be the number of small eigenvalues in $\{\lambda_{M,i}\}$. According to [Pottmann 1999], the segment has f_r rotation free directions. With both f_t and f_r, surfaces can be classified into 4 groups as shown in Table 6.1. Practically, an eigenvalue is considered to be small when one of the following conditions are true:

$$\begin{aligned} \lambda_{L,i} &< 0.05 \\ \lambda_{M,i} &< 0.5\tau^2 \end{aligned} \tag{6.22}$$

6.4.2 Workflow

Let us take one segment of SoR as an example to illustrate the workflow. All other segments follow the same procedure.

Two sets of surfels of the SoR, S^m and S^v, can be obtained from [Tateno 2015] at each frame. S^m contains all surfels in the model obtained from the SLAM procedure. S^v is the visible subset of S^m at current frame. When the SoR has only appeared in front of the camera for a few frames, or when it is occluded or far from the camera, the acquired data may be not stable. With more information coming from new frames, data become more stable and the reconstruction more reliable. We use a confidence label c to represent the stability of data points acquired from the SoR.

The workflow to reconstruct the SoR is shown in Fig. 6.4. When c is less than a

threshold c_t, we check the stability by estimating the variation of S^v's rotation axis. Although different frames contain different sets of visible surfels, their rotation axis should be similar. Thus, c is incremented by 1 if the angle between two estimations of the rotation axis is smaller than a threshold γ_t (set to 10° in our experiment). When $c = c_t$ (set to 5 in our experiment), the data is considered to be stable. The rotation axis of S^m is estimated and the SoR is then reconstructed for the first time by applying our methods (cf. Sections 6.2-6.3) on the global model S^m, with τ (cf. Section 6.3.2) being the average of diameters (i.e. 2×radius) of all surfels in S^m. More details are shown in Alg. 12.

Algorithm 12 Stability Estimation and first reconstruction

Input: S^v, S^m at current frame
Output: (Possibly) first reconstruction
Calculate $\vec{l}$ from S^v using P-method [Pottmann 1999]
if $c = 0$ **then**
  Set $\vec{l}^r = \vec{l}$
else if $\cos^{-1}(\vec{l} \cdot \vec{l}^r) < \gamma_t$ **then**
  $c = c + 1$
  if $c \geq c_t$ **then**
    Estimate initial $\vec{l}_0^m$ from S^m with P-method
    Improve axis estimation & reconstruct from S^m
else
  $c = c - 1$

In later frames, we continue to check the precision of this reconstruction. If the reconstruction is precise, the reconstructed model should fit well all visible surfels S^v in new frames. Otherwise, the rotation axis of the SoR should be re-estimated and the SoR should be reconstructed by using the newest global model S^m. Details on checking the reconstruction precision are shown in Alg. 13, where *enum* indicates the number of surfels whose modeling error is larger than e_t. Empirically, e_t is set to $1.5e$, with e being the SoR's global reconstruction error (cf. Section 6.3.1). When *enum* is larger than a threshold N_t, the reconstruction should be improved. Since consecutive frames have large overlapping area, the same surfel may be frequently recalculated. We found that $N_t = 30\%|S^m|$ works well in our experiment.

The *calcModelingError* process estimates the modeling error e_i on surfel s_i. e_i is computed as the distance between s_i and its approximate nearest point on the reconstructed surface. The approximate nearest point is constrained so that itself and s_i are lying on the same plane which is perpendicular to the rotation axis.

Algorithm 13 Check reconstruction precision and improve reconstruction

Input: S^v, S^m at current frame, previous reconstruction
Output: (Possibly) new reconstruction
for each $s_i \in S^v$ **do**
  $e_i \leftarrow calcModelingError(s_i, S^m)$
  $enum = enum + 1$ if $e_i > e_t$
if $enum > N_t$ **then**
  Improve axis estimation and reconstruct the SoR from S^m
  Set $enum = 0$

6.5 Results

In this section, we first evaluate the method for estimating the rotation axis, before using real data to evaluate reconstruction results and the efficiency of our method.

6.5.1 Synthetic study

We use synthetic truncated cone to investigate the feasibility of the method (cf. Fig. 6.5). The cone is created with small radius $r = 20$, height $h = 10$ and grid size $\tau = 0.2$. Since we only calculate the error on the rotation axis estimation, $n_1 = 10$ is used.

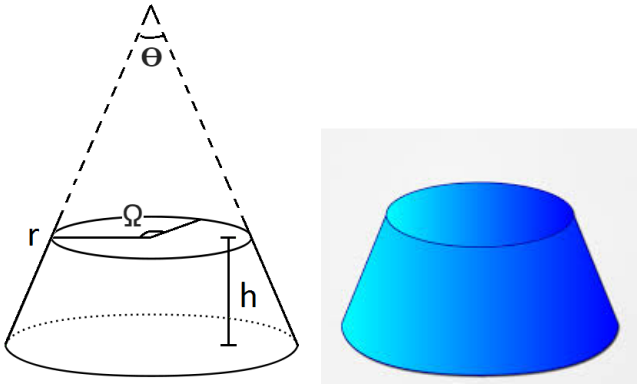


Figure 6.5: A synthetic truncated cone: with small radius r, height h, angular span Ω, and the opening angle Θ.

To simulate noise, $\delta_r \sim N(0, 0.5\tau)$ is added to the three coordinates of each point of the synthetic truncated cone, and $\delta_n \sim N(0, \eta)$ is added to each normal component (normalized afterwards). At last, the synthetic cone is rotated and moved by a random rigid transformation. We compare the results of our method with the one from [Pottmann 1999] (denoted as *P-method*), since we are interested in efficient methods for real-time application.

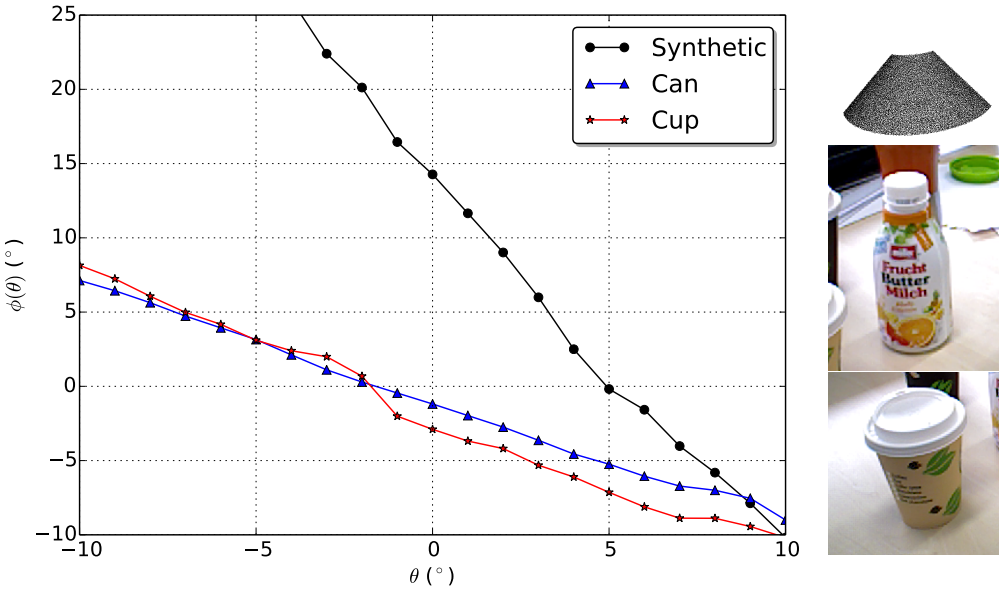


Figure 6.6: $\phi(\theta)$ for different objects: synthetic cone (right top), Can (right middle) and Cup (right bottom). They are approximately linear.

The objective functions $\phi(\theta)$ of different objects are shown in Fig. 6.6. For the synthetic cone, we use $\Theta = 100°$, $\eta = 0.03$. The Cup and Can objects are taken from Dataset 2 of Section 6.5.2 (cf. Fig. 6.10). We can see that these $\phi(\theta)$s are almost linear, when θ is near the ground truth.

Fig. 6.7 shows the rotation axis estimation error $\cos^{-1}(\vec{l}_g, \vec{l})$ on synthetic data. Our method gives better results, especially for large noise and opening angles. Moreover, the error of our method is less than 0.5° in most cases.

6.5.2 Real data

For real-time reconstruction, $n_1 = n_2 = 10$ is used. Two datasets are used for evaluating our online reconstruction algorithm. We use downsampled images to resolution 160×120 to let the whole workflow run in real-time on the CPU as indicated in [Tateno 2015].

We use two cylinders and a cone with known size in Dataset 1. The cylinders' radii and the cone's opening angle are used to measure the precision of the algorithm. Results are shown in Fig. 6.8. It can be seen that the accuracy of cylinders' radii is about 1-2mm, while the accuracy of the cone's opening angle has a big variation at the beginning but gradually converges near the ground truth in about 80 frames.

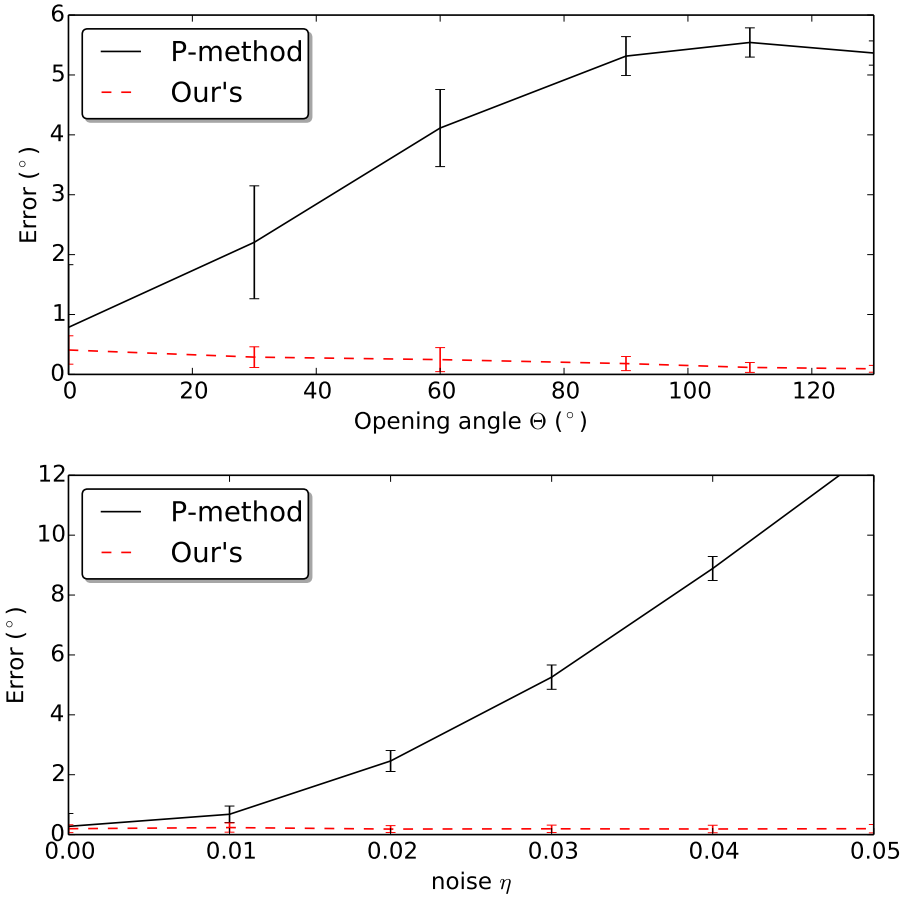


Figure 6.7: Synthetic result: errors on the direction of the rotation axis. Top: errors with respect to opening angles Ω, with $\eta = 0.03$; Bottom: errors with respect to the amount of noise η, with $\Omega = 120°$.

Dataset 2 is an online dataset provided by Tateno[1]. It contains diverse daily necessities lying on a desktop. We draw the time consumption for each frame in Fig. 6.9. The solid line represents the reconstruction time, with an average of 1.87ms per frame, while the dashed line represents the SLAM+segmentation time, with an average of 20.40ms per frame. Therefore, on average our method runs at 45fps. These results are produced on a laptop equipped with Intel Core i7-4510U CPU @2.00GHz and 8GB RAM.

The final reconstructed objects along with the segmentation map is presented in Fig. 6.10. The reconstructed objects correctly match the original data and are visually similar to the original objects in the RGB image.

[1] http://campar.in.tum.de/Chair/ProjectInSeg

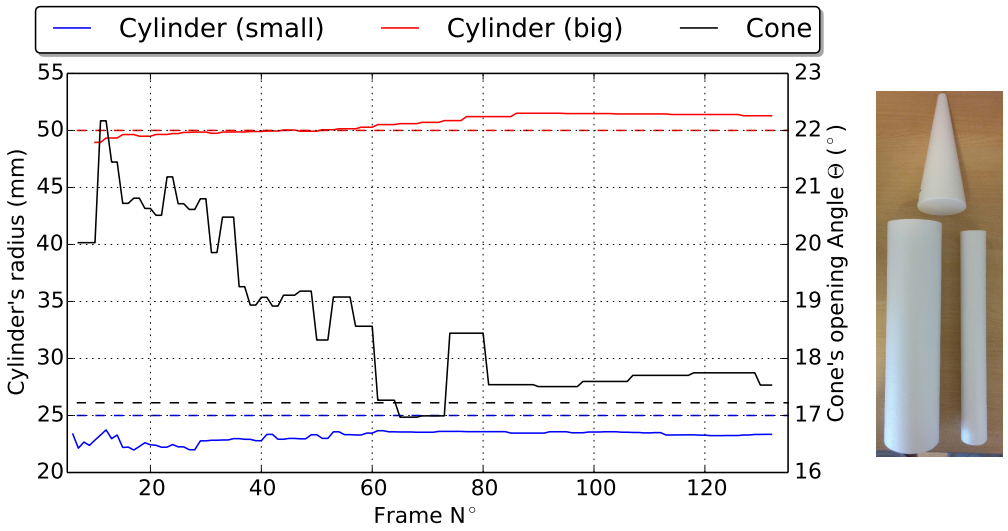


Figure 6.8: Dataset 1: Measuring objects of known dimension. Dashed lines represent ground truth values. Solid lines represent measurements from reconstruction results.

6.6 Conclusion

We have developed a fast and accurate method to estimate the rotation axis of revolution surfaces as well as a real-time online geometric reconstruction workflow. Experiments show that the proposed method and workflow work well for real world objects, achieving about 1-2mm accuracy for radius estimation and $< 1°$ for cone's opening angle estimation.

With these reconstructed geometries, point clouds can be simplified and other interesting work may be performed. For example, SLAM wide-baseline re-localization problems could be solved by 3D rotation axis matching. Other future work may also include improving segmentation results by using reconstructed geometries; establishing spatial relationships between different geometries, etc.

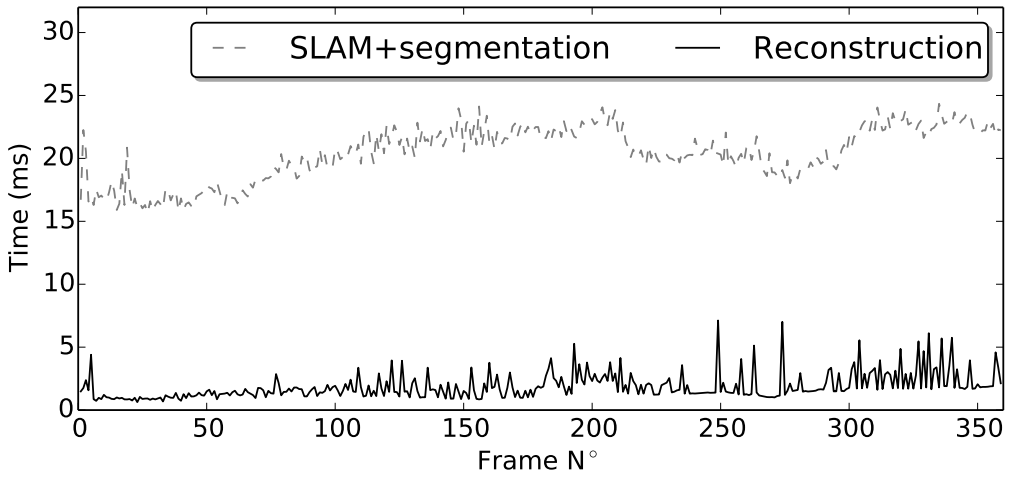


Figure 6.9: Dataset 2: Time consumption.

Figure 6.10: Dataset 2. Top: Original segmented point cloud (left) and our reconstruction result (right). Bottom: visual comparison between RGB image (left), original point cloud (middle) and reconstruction result (right) for Cup and Can objects.

Chapter 7

Conclusion

Contents

This dissertation was motivated by the drawbacks of texture-based wide baseline localization (i.e. tracking by detection) methods. It focuses on geometry-based methods, which can be used when targets contain little or no texture, or lighting conditions change a lot. Such methods can also be used to register targets under different representations, such as graphs vs. images. Results of this work show that wide base line localization can be performed by means of Point Pattern Matching (PPM). The major contributions of this dissertations are:

7.1 Point Pattern Matching algorithms

Chapters 3 and 4 presented two algorithms for PPM, i.e. Robust Random Dot Markers (RRDM) and Local Geometric Consensus (LGC). Both of them are robust against acquisition noise, such as over/under detected points, jitter, as well as partial occlusions.

7.1.1 2D point pattern matching

RRDM aims at solving registration problems between two point sets undergoing 2D perspective transformations. It splits a whole point set into small local subsets to avoid the huge computational complexity. A robust Two-Surface-Ratio (TSR) descriptor is designed to describe spatial relationships between points, which remains

stable even with the presence of point jitter. A local voting scheme is used for quick local matchings. Moreover, RRDM proposed for the first time to use geometric transformation consensus between neighboring subsets to reject outliers. Analysis and synthetic experiments show that RRDM has a quadratic behavior with respect to the number of points.

Contribution: RRDM demonstrates that PPM can be both robust and fast enough for augmented reality applications. Unprepared paper maps of different colors can be registered with the same data extracted from a Geographic Information System (GIS) by relying on road intersections. Although the quality of the automatically detected road intersections from paper maps can be very poor, a correct registration between the GIS and the map is still possible with the help of RRDM.

7.1.2 LGC: a general solution for PPM

LGC tackles the problem of multi point sets recognition and matching in a general way. A hypotheses generator is used to propose putative correspondences of patches (subset of points). A hypotheses validator is used to identify whether these putative correspondences are inliers by trying to find other correspondences in their neighborhood. A result refiner is finally used to find more correspondences in order to give more precise results. Compared to RRDM, LGC employs Geometric Hashing instead of TSR descriptors to match subsets so that the algorithm can be adapted to transformations in any dimension theoretically. LGC makes better use of consensus between local subsets than RRDM. The concept of local geometric consensus helps to distinguish inliers from outliers, as well as to judge the presence of a model point set with low false alert rate. Both analysis and experiments show that LGC has a linear behavior relative to the number of points, which is much more efficient than than classic Geometric Hashing and RRDM. Compared to RDM, LGC is much more robust and even a bit faster in some challenging cases.

Contribution: LGC is a general algorithm for PPM, which can deal with different types of transformations in different dimensions. It demonstrates that PPM can be both robust and fast enough for real-time tracking to be used in augmented reality applications. Different types of targets can be modeled as point patterns and can be recognized and tracked by LGC. As to texture-less objects, LGC outperforms traditional methods such as SURF.

7.2 Projector-camera system calibration

The projector-camera system is used in Spatial Augmented Reality (SAR). In large scale SAR, the projector's is usually set at large distance and hence projections at short distance are defocused. This makes the projector-camera system difficult to calibrate under such situations. People either need to use a huge calibration board, acquire a precise 3D model, or wait several minutes for the calibration using traditional methods. An application for calibrating a projector-camera system is developed to solve this problem. It relies on two random point patterns and LGC for calibration. The application is practical and calibrates accurately whatever the projector's focus distance. The user only needs to manipulate a B4 size (250×353mm) rigid board during the calibration. The superiority of the method relies on the fact that (1) In a defocused situation, dots can be more easily detected than targets requiring edges information (such as chessboards, ARToolKit markers, etc.). (2) Random dots can be robustly tracked by LGC.

Contribution: The proposed application is practical and as precise as one representative state of art. It produces calibration results that are stable and accurate with a small calibration board whatever the projector's focus distance. For example, the reprojection error is only about 4mm for a 4.5m projection distance, which is comparable to the state of art method using a 22 times larger calibration board.

7.3 Scene understanding in 3D data

Chapter 6 presented a piece of work related to scene understanding from 3D data. It proposed a novel method to estimate the rotation axis of a Surface of Revolution (SoR) efficiently. It also designed a workflow to integrate real-time SoR reconstruction into dense SLAM. By taking advantage of the symmetry of SoR, it proves that the rotation axis of SoR can be found using a one dimension search. An approximate objective function is also proposed to obtain fast convergences.

Contribution: A fast method to estimate the rotation axis of SoR is proposed, which makes the real-time reconstruction of SoR possible. The work can provide basic elements for high level structure modeling of the scene.

7.4 Perspectives

Although some interesting applications have been developed, there are still many improvements that can be considered to make them more useful. Regarding the

projector-camera calibration application, a fully automated method would be of interest in the future and could be achieved by using a non planar calibration target. Experiments on projectors with extreme focal lengths (e.g. very short/long focal lengths) are also interesting research topics. For the augmented map application, a road layer extraction which does not need any user interaction can be beneficial in order to create a fully automated application. Quick map localization in GIS which contains a much wider area than the map is also an interesting topic (cf. [Minster 2015]). As to real-time reconstruction of SoR, more experiments can be performed to study the sensitivity of the method with respect to acquisition noise. More comparisons against other existing but slower methods are needed to study in detail whether our real-time solution has an impact on the precision of reconstructions. Real-time reconstruction of more generic primitives, such as superquadric primitives, is also an interest of research in the future.

As to LGC, two directions can be exploited, i.e. improvements of the algorithm and development of new applications.

Influence on the performance of some parameters is not fully studied, such as k (i.e. the number of nearest neighbors). More comprehensive studies on these parameters are required to find their optimal values in different cases, which would help the algorithm to achieve better performance. Similarly, parameter settings on other kinds of transformation can also be studied. The algorithm can be improved to better deal with partially regular point patterns, for example, by selecting carefully points that are fed to the generator. Similar to the "colored LGC" in Section 4.3.3, other ways of combining both geometric information and "color" information for registration can also be exploited in order to gain performance when points have their own characteristics. The problem of massive extra or missing points is another challenge to be faced as well. It often happens when two point patterns to be matched are detected from the same target in different scales. For example, lots of details may be lost in an image with a scale of lower resolution, and thus it produces massive missing points. The current LGC cannot solve this problem.

Since LGC is a point pattern matching algorithm, which is an elementary task in many problems, it may have many other potential applications. Current monocular SLAM techniques rely on texture-based methods to establish point correspondences, and such methods fail if they work in texture-less environments, such as construction sites. By using LGC as the point matching module, texture-less monocular SLAM may be doable. The registration between a scanned 3D model (i.e. a dense point cloud) and its corresponding CAD model (e.g. triangle meshes with sparse points) is difficult. It could also be solved by LGC if 3D geometric salient

points can be found from both models, for example, by using Support Vector Shape (SVS) [Van Nguyen 2013]. Similarly, a solution to re-localizations of dense SLAM in case of tracking failure may also be proposed based on LGC. Other transformation types can further be considered. By applying mapping functions such as *equisolid angle*, one may be able to match a planar object with its image captured by a fisheye camera.

Besides LGC and point pattern matchings, other related research directions are interesting as well in the future. Since LGC needs reliable and repeatable point patterns as inputs, studies on stable and repeatable point detectors will be very beneficial. As to texture-less tracking, LGC could be integrated with Virtual Visual Servoing (VVS) [Comport 2006]. New methods on texture-less wide baseline localization can also be developed by using other types of geometries, such as regions and line segments, together with points. Or, a combination of different texture-less registration methods may be interesting in order to propose a general solution.

Bibliography

[Agrawal 1994] Ashish Agrawal, Nirwan Ansari and Edwin SH Hou. *Evolutionary programming for fast and robust point pattern matching*. In Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on, volume 3, pages 1777–1782. IEEE, 1994. (Cited on page 24.)

[Amit 1997] Yali Amit and Donald Geman. *Shape quantization and recognition with randomized trees*. Neural computation, vol. 9, no. 7, pages 1545–1588, 1997. (Cited on page 17.)

[Audet 2009] Samuel Audet and Masatoshi Okutomi. *A user-friendly method to geometrically calibrate projector-camera systems*. In IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 47–54, 2009. (Cited on pages xxi, xxii, 94, 95, 103, 104, 106 and 112.)

[Awrangjeb 2012] Mohammad Awrangjeb, Guojun Lu and Clive S Fraser. *Performance comparisons of contour-based corner detectors*. IEEE Transactions on Image Processing, vol. 21, no. 9, pages 4167–4179, 2012. (Cited on page 15.)

[Azuma 1997] Ronald T Azuma. *A survey of augmented reality*. Presence: Teleoperators and virtual environments, vol. 6, no. 4, pages 355–385, 1997. (Cited on pages 1, 3 and 13.)

[Azuma 2001] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier and Blair MacIntyre. *Recent advances in augmented reality*. IEEE computer graphics and applications, vol. 21, no. 6, pages 34–47, 2001. (Cited on page 13.)

[Ballard 1981] Dana H. Ballard. *Generalizing the Hough transform to detect arbitrary shapes*. Pattern Recogn., vol. 13, no. 2, pages 111–122, January 1981. (Cited on page 22.)

[Ballesta 2008] Mónica Ballesta, Arturo Gil, Oscar Reinoso and O Martinez Mozos. *Evaluation of interest point detectors for visual SLAM*. International Journal of Factory Automation, Robotics and Soft Computing, vol. 4, pages 86–95, 2008. (Cited on page 16.)

[Bay 2005] Herbert Bay, Vittorio Ferraris and Luc Van Gool. *Wide-baseline stereo matching with line segments.* In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 1, pages 329–336. IEEE, 2005. (Cited on pages 16 and 17.)

[Bay 2008] Herbert Bay, Andreas Ess, Tinne Tuytelaars and Luc Van Gool. *Speeded-Up Robust Features (SURF).* Comput. Vis. Image Underst., vol. 110, no. 3, pages 346–359, June 2008. (Cited on pages 8, 15 and 16.)

[Beis 1997] Jeffrey S Beis and David G Lowe. *Shape indexing using approximate nearest-neighbour search in high-dimensional spaces.* In Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on, pages 1000–1006. IEEE, 1997. (Cited on page 17.)

[Besl 1992a] P. J. Besl and H. D. McKay. *A method for registration of 3-D shapes.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, pages 239–256, Feb 1992. (Cited on page 115.)

[Besl 1992b] Paul J. Besl and Neil D. McKay. *A Method for Registration of 3-D Shapes.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 14, no. 2, pages 239–256, February 1992. (Cited on page 22.)

[Billinghurst 2015] Mark Billinghurst, Adrian Clark and Gun Lee. *A survey of augmented reality.* Foundations and Trends in Human-Computer Interaction, vol. 8, no. 2-3, pages 73–272, 2015. (Cited on pages 1 and 2.)

[Bimber 2005] Oliver Bimber and Ramesh Raskar. Spatial Augmented Reality: Merging Real and Virtual Worlds. CRC Press, 2005. (Cited on pages xvii, 3 and 4.)

[Botterill 2009] Tom Botterill, Steven Mills and Richard D Green. *New Conditional Sampling Strategies for Speeded-Up RANSAC.* In BMVC, pages 1–11. Citeseer, 2009. (Cited on page 20.)

[Bradski 2008] Gary Bradski and Adrian Kaehler. Learning opencv. O'Reilly, 2008. (Cited on page 96.)

[Caetano 2006] Tiberio S Caetano, Terry Caelli, Dale Schuurmans and Dante Augusto Couto Barone. *Graphical models and point pattern matching.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 10, pages 1646–1663, 2006. (Cited on page 24.)

[Callier 2011] Sébastien Callier and Hideo Saito. *Automatic Road Extraction from Printed Maps.* In IAPR Conference on Machine Vision Application, Nara, Japan, 2011. (Cited on pages 10, 26 and 48.)

[Callier 2012] Sebastien Callier and Hideo Saito. *Automatic Road Area Extraction from Printed Maps Based on Linear Feature Detection.* IEICE Transactions, pages 1758–1765, 2012. (Cited on page 51.)

[Calonder 2012] Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha and Pascal Fua. *BRIEF: Computing a Local Binary Descriptor Very Fast.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 7, pages 1281–1298, July 2012. (Cited on pages 8, 16 and 17.)

[Campbell 2015] Dylan Campbell and Lars Petersson. *An adaptive data representation for robust point-set registration and merging.* In Proceedings of the IEEE International Conference on Computer Vision, pages 4292–4300, 2015. (Cited on page 18.)

[Canny 1986] John Canny. *A computational approach to edge detection.* IEEE Transactions on pattern analysis and machine intelligence, no. 6, pages 679–698, 1986. (Cited on page 52.)

[Chan 2016] Jacob Chan, Jimmy Addison Lee and Qian Kemao. *BORDER: An Oriented Rectangles Approach to Texture-Less Object Recognition.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2855–2863, 2016. (Cited on page 19.)

[Chiang 2011] Yao-Yi Chiang and Craig Knoblock. *A general approach for extracting road vector data from raster maps.* International Journal on Document Analysis and Recognition (IJDAR), vol. 16, no. 1, pages 55–81, October 2011. (Cited on page 52.)

[Choi 2006] Vicky Choi and Navin Goyal. *An efficient approximation algorithm for point pattern matching under noise.* In Latin American Symposium on Theoretical Informatics, pages 298–310. Springer, 2006. (Cited on page 10.)

[Choi 2012] Changhyun Choi and Henrik I Christensen. *3D textureless object detection and tracking: An edge-based approach.* In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3877–3884. IEEE, 2012. (Cited on pages 7 and 18.)

[Chui 2000] Haili Chui and Anand Rangarajan. *A new algorithm for non-rigid point matching*. In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, volume 2, pages 44–51. IEEE, 2000. (Cited on page 23.)

[Chum 2005] O. Chum and J. Matas. *Matching with PROSAC - Progressive Sample Consensus*. In Proc. of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '05, pages 220–226, 2005. (Cited on pages 17, 20 and 38.)

[Coiras 2000] Enrique Coiras, Javier Santamarı, Carlos Miravet*et al.* *Segment-based registration technique for visual-infrared images*. Optical Engineering, vol. 39, no. 1, pages 282–289, 2000. (Cited on page 19.)

[Comport 2006] Andrew I Comport, Eric Marchand, Muriel Pressigout and Francois Chaumette. *Real-time markerless tracking for augmented reality: the virtual visual servoing framework*. IEEE Transactions on visualization and computer graphics, vol. 12, no. 4, pages 615–628, 2006. (Cited on pages 7 and 137.)

[Cortes 1995] Corinna Cortes and Vladimir Vapnik. *Support-vector networks*. Machine learning, vol. 20, no. 3, pages 273–297, 1995. (Cited on page 18.)

[Damen 2011] Dima Damen, Andrew Gee, Andrew Calway and Walterio Mayol-Cuevas. *Detecting and localising multiple 3D objects: A fast and scalable approach*. In IROS Workshop on Active Semantic Perception and Object Search in the Real World (ASP-AVS-11), pages 1–6, 2011. (Cited on page 19.)

[Datta 2009] Ankur Datta, Jun-Sik Kim and Takeo Kanade. *Accurate Camera Calibration using Iterative Refinement of Control Points*. In IEEE 12th International Conference on Computer Vision Workshops, pages 1201–1208, 2009. (Cited on page 98.)

[Datta 2013] Abhik Datta, Adams W.-K. Kong, Soumita Ghosh and Dieter Trau. *A fast point pattern matching algorithm for robust spatially addressable bead encoding*. In Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference on, pages 1–7, Nov 2013. (Cited on pages xvii, 11, 20 and 22.)

[Denton 2007] Jason A Denton and J Ross Beveridge. *An algorithm for projective point matching in the presence of spurious points*. Pattern recognition, vol. 40, no. 2, pages 586–595, 2007. (Cited on page 20.)

[Donoser 2006] Michael Donoser and Horst Bischof. *Efficient maximally stable extremal region (MSER) tracking.* In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), volume 1, pages 553–560. IEEE, 2006. (Cited on page 15.)

[Donoser 2011] Michael Donoser, Peter Kontschieder and Horst Bischof. *Robust planar target tracking and pose estimation from a single concavity.* In Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on, pages 9–15. IEEE, 2011. (Cited on page 18.)

[Drost 2015] Bertram Drost and Slobodan Ilic. *Local Hough Transform for 3D Primitive Detection.* In 3D Vision (3DV), 2015 International Conference on, pages 398–406. IEEE, 2015. (Cited on page 114.)

[Duane 1971] C BROWN Duane. *Close-range camera calibration.* Photogramm. Eng, vol. 37, no. 8, pages 855–866, 1971. (Cited on page 6.)

[Ezoji 2006] Mehdi Ezoji, FAEZ Karim, Hamidreza Rashidy Kanan and Saeed Mozaffari. *GA-based affine PPM using matrix polar decomposition.* IEICE transactions on information and systems, vol. 89, no. 7, pages 2053–2060, 2006. (Cited on page 24.)

[Fiala 2005] Mark Fiala. *ARTag, a fiducial marker system using digital techniques.* In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 2, pages 590–596. IEEE, 2005. (Cited on pages xvii and 14.)

[Filipe 2014] Silvio Filipe and Luís A. Alexandre. *A Comparative Evaluation of 3D Keypoint Detectors in a RGB-D Object Dataset.* In VISAPP 2014 - Proc. of the 9th International Conference on Computer Vision Theory and Applications, pages 476–483, 2014. (Cited on page 80.)

[Fischler 1981] Martin A. Fischler and Robert C. Bolles. *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography.* Commun. ACM, vol. 24, no. 6, pages 381–395, June 1981. (Cited on page 20.)

[Fitzgibbon 2003] Andrew W Fitzgibbon. *Robust registration of 2D and 3D point sets.* Image and Vision Computing, vol. 21, no. 13, pages 1145–1153, 2003. (Cited on page 22.)

[Flusser 1994] Jan Flusser and Tomas Suk. *A moment-based approach to registration of images with affine geometric distortion.* IEEE transactions on Geoscience and remote sensing, vol. 32, no. 2, pages 382–387, 1994. (Cited on page 18.)

[Forssén 2007] Per-Erik Forssén and David G Lowe. *Shape descriptors for maximally stable extremal regions.* In 2007 IEEE 11th International Conference on Computer Vision, pages 1–8. IEEE, 2007. (Cited on pages 15 and 16.)

[Gao 2008] Wei Gao, Liang Wang and Zhanyi Hu. *Flexible method for structured light system calibration.* Optical Engineering, vol. 47, no. 8, pages 083602–083602, 2008. (Cited on pages xxi and 94.)

[Gatrell 1992] Lance B. Gatrell, William A. Hoff and Cheryl W. Sklair. *Robust image features: concentric contrasting circles and their image extraction.* In Proceedings of SPIE, Cooperative Intelligent Robotics in Space II, volume 1612, pages 235–244, 1992. (Cited on page 14.)

[Gennery 1992] Donald B Gennery. *Visual tracking of known three-dimensional objects.* International Journal of Computer Vision, vol. 7, no. 3, pages 243–270, 1992. (Cited on page 7.)

[Gold 1998] Steven Gold, Anand Rangarajan, Chien-Ping Lu, Suguna Pappu and Eric Mjolsness. *New algorithms for 2D and 3D point matching: pose estimation and correspondence.* Pattern Recogn., vol. 31, no. 8, pages 1019–1031, August 1998. (Cited on page 23.)

[González 2005] J Isern González, J Cabrera Gámez, C Guerra Artal and AM Naranjo Cabrera. *Stability Study of Camera Calibration Methods.* In CI Workshop en Agentes Físicos, WAF, 2005. (Cited on page 104.)

[Goshtasby 1986] Ardeshir Goshtasby, George C Stockman and Carl V Page. *A region-based approach to digital image registration with subpixel accuracy.* IEEE Transactions on Geoscience and Remote Sensing, no. 3, pages 390–399, 1986. (Cited on page 18.)

[Granger 2002] Sébastien Granger and Xavier Pennec. *Multi-scale EM-ICP: A fast and robust approach for surface registration.* In European Conference on Computer Vision, pages 418–432. Springer, 2002. (Cited on page 23.)

[Grompone 2010] von Gioi R Grompone, Jeremie Jakubowicz, Jean-Michel Morel and Gregory Randall. *LSD: a fast line segment detector with a false detection*

control. IEEE transactions on pattern analysis and machine intelligence, vol. 32, no. 4, pages 722–732, 2010. (Cited on page 15.)

[Gros 1998] Patrick Gros, Olivier Bournez and Edmond Boyer. *Using local planar geometric invariants to match and model images of line segments.* Computer Vision and Image Understanding, vol. 69, no. 2, pages 135–155, 1998. (Cited on page 19.)

[Gruen 2002] A Gruen*et al.* *Calibration and orientation of cameras in computer vision*, 2002. (Cited on page 7.)

[Guan 2009] Wei Guan, Lu Wang, Jonathan Mooser, Suya You and Ulrich Neumann. *Robust pose estimation in untextured environments for augmented reality applications.* In Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on, pages 191–192. IEEE, 2009. (Cited on page 19.)

[Hagbi 2009] Nate Hagbi, Oriel Bergig, Jihad El-Sana and Mark Billinghurst. *Shape recognition and pose estimation for mobile augmented reality.* In Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on, pages 65–71. IEEE, 2009. (Cited on page 18.)

[Han 2012] Dongjin Han, David B Cooper and Hern-soo Hahn. *Fast axis estimation from a segment of rotationally symmetric object.* In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 1154–1161. IEEE, 2012. (Cited on page 115.)

[Harris 1988] Chris Harris and Mike Stephens. *A combined corner and edge detector.* In Alvey vision conference, volume 15, page 50. Citeseer, 1988. (Cited on pages 15 and 16.)

[Hartley 2004] Richard I. Hartley and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge University Press, ISBN: 0521540518, second édition, 2004. (Cited on pages 31, 44, 45 and 99.)

[Heyl 2013] Jeremy S Heyl. *kd Match: A Fast Matching Algorithm for Sheared Stellar Samples.* arXiv preprint arXiv:1304.0838, 2013. (Cited on page 21.)

[Hilliges 2012] Otmar Hilliges, David Kim, Shahram Izadi, Malte Weiss and Andrew Wilson. *HoloDesk: direct 3d interactions with a situated see-through display.* In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 2421–2430. ACM, 2012. (Cited on pages xvii and 3.)

[Holzer 2009] Stefan Holzer, Stefan Hinterstoisser, Slobodan Ilic and Nassir Navab. *Distance transform templates for object detection and pose estimation.* In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 1177–1184. IEEE, 2009. (Cited on pages 8, 18 and 19.)

[Huttenlocher 1992] Daniel P Huttenlocher and William J Rucklidge. *A multi-resolution technique for comparing images using the Hausdorff distance.* Rapport technique, Cornell University, 1992. (Cited on page 22.)

[Huynh 2009] Du Q. Huynh. *Metrics for 3D Rotations: Comparison and Analysis.* J. Math. Imaging Vis., vol. 35, no. 2, pages 155–164, October 2009. (Cited on pages 45 and 74.)

[Jian 2011] Bing Jian and Baba C Vemuri. *Robust point set registration using gaussian mixture models.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 8, pages 1633–1645, 2011. (Cited on page 23.)

[Jones 2001] MC Jones, Nils Lid Hjort, Ian R Harris and Ayanendranath Basu. *A comparison of related density-based minimum divergence estimators.* Biometrika, vol. 88, no. 3, pages 865–873, 2001. (Cited on pages 22 and 23.)

[Jones 2014] Brett Jones, Rajinder Sodhi, Michael Murdock, Ravish Mehra, Hrvoje Benko, Andrew Wilson, Eyal Ofek, Blair MacIntyre, Nikunj Raghuvanshi and Lior Shapira. *RoomAlive: magical experiences enabled by scalable, adaptive projector-camera units.* In Proceedings of the 27th annual ACM symposium on User interface software and technology, pages 637–644. ACM, 2014. (Cited on pages xvii and 3.)

[Jurie 2001] Frédéric Jurie and Michel Dhome. *A simple and efficient template matching algorithm.* In Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, volume 2, pages 544–549. IEEE, 2001. (Cited on page 7.)

[Kato 1999] H. Kato and M. Billinghurst. *Marker tracking and HMD calibration for a video-based augmented reality conferencing system.* Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99), pages 85–94, 1999. (Cited on pages xvii, 3, 8 and 14.)

[Lamdan 1990] Yehezkel Lamdan, Jacob T Schwartz and Haim J Wolfson. *Affine Invariant Model-Based Object Recognition.* IEEE Trans. Robot. Autom., vol. 6, no. 5, pages 578–589, October 1990. (Cited on page 73.)

[Lamdan 1991] Yehezkel Lamdan and Haim J Wolfson. *On the error analysis ofgeometric hashing'.* In Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on, pages 22–27. IEEE, 1991. (Cited on page 21.)

[Lang 2010] Dustin Lang, David W Hogg, Keir Mierle, Michael Blanton and Sam Roweis. *Astrometry. net: Blind astrometric calibration of arbitrary astronomical images.* The astronomical journal, vol. 139, no. 5, page 1782, 2010. (Cited on pages xvii, 11, 20 and 21.)

[Lepetit 2005] Vincent Lepetit and Pascal Fua. *Monocular Model-based 3D Tracking of Rigid Objects.* Foundations and Trends in Computer Graphics and Vision, vol. 1, no. 1, pages 1–89, January 2005. (Cited on pages 6, 7 and 14.)

[Li 2008] Zhongwei Li, Yusheng Shi, Congjun Wang and Yuanyuan Wang. *Accurate calibration method for a structured light system.* Optical Engineering, vol. 47, no. 5, pages 053604–053604, 2008. (Cited on page 93.)

[Li 2014] Beiwen Li, Nikolaus Karpinsky and Song Zhang. *Novel calibration method for structured-light system with an out-of-focus projector.* Applied optics, vol. 53, no. 16, pages 3415–3426, 2014. (Cited on pages xxi, 93 and 94.)

[Li 2016] Kai Li, Jian Yao, Mengsheng Lu, Yuan Heng, Teng Wu and Yinxuan Li. *Line segment matching: A benchmark.* In 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 1–9. IEEE, 2016. (Cited on page 19.)

[Long 2014] Tengfei Long, Weili Jiao, Guojin He and Wei Wang. *Automatic line segment registration using Gaussian mixture model and expectation-maximization algorithm.* IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 7, no. 5, pages 1688–1699, 2014. (Cited on page 19.)

[Lou 2009] C Lou, L Zhu and H Ding. *Identification and reconstruction of surfaces based on distance function.* Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, vol. 223, no. 8, pages 981–994, 2009. (Cited on pages 115 and 126.)

[Lowe 1999] David G Lowe. *Object recognition from local scale-invariant features.* In Computer vision, 1999. The proceedings of the seventh IEEE international

conference on, volume 2, pages 1150–1157. Ieee, 1999. (Cited on pages xvii, 15, 16 and 17.)

[Lowe 2004] David G. Lowe. *Distinctive Image Features from Scale-Invariant Keypoints.* Int. J. Comput. Vision, vol. 60, no. 2, pages 91–110, November 2004. (Cited on pages 8, 17 and 20.)

[Luo 2014] Huafen Luo, Jing Xu, Nguyen Hoa Binh, Shuntao Liu, Chi Zhang and Ken Chen. *A simple calibration procedure for structured light system.* Optics and Lasers in Engineering, vol. 57, pages 6–12, 2014. (Cited on page 92.)

[Magic Leap 2015] Inc. Magic Leap. *Magic Leap Demo.* https://www.youtube.com/watch?v=kw0-JRa9n94, 2015. [Online; accessed 02-August-2016]. (Cited on pages xvii and 3.)

[Maimone 2013] Andrew Maimone, Xubo Yang, Nate Dierk, Andrei State, Mingsong Dou and Henry Fuchs. *General-purpose telepresence with head-worn optical see-through displays and projector-based lighting.* In 2013 IEEE Virtual Reality (VR), pages 23–26. IEEE, 2013. (Cited on pages xvii and 3.)

[Mara 2006] Hubert Mara and Robert Sablatnig. *Orientation of Fragments of Rotationally Symmetrical 3D-Shapes for Archaeological Documentation.* In 3D Data Processing, Visualization, and Transmission, Third International Symposium on, pages 1064–1071. IEEE, 2006. (Cited on pages 115 and 116.)

[Marchand 2016] E. Marchand, H. Uchiyama and F. Spindler. *Pose estimation for augmented reality: a hands-on survey.* IEEE Transactions on Visualization and Computer Graphics, vol. PP, no. 99, pages 1–1, 2016. (Cited on page 13.)

[Martedi 2013] Sandy Martedi, Bruce Thomas and Hideo Saito. *Region-based tracking using sequences of relevance measures.* In Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on, pages 1–6. IEEE, 2013. (Cited on page 18.)

[McIlroy 2012] Paul McIlroy, Shahram Izadi and Andrew Fitzgibbon. *Kinectrack: Agile 6-DoF Tracking Using a Projected Dot Pattern.* In Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on, pages 23–29, 2012. (Cited on page 86.)

[Milgram 1995] Paul Milgram, Haruo Takemura, Akira Utsumi and Fumio Kishino. *Augmented reality: A class of displays on the reality-virtuality continuum.* In

Photonics for industrial applications, pages 282–292. International Society for Optics and Photonics, 1995. (Cited on pages xvii, 1 and 2.)

[Minster 2015] Gautier Minster, Guillaume Moreau and Hideo Saito. *Geolocation for printed maps using line segment-based SIFT-like feature matching.* In Mixed and Augmented Reality Workshops (ISMARW), 2015 IEEE International Symposium on, pages 88–93. IEEE, 2015. (Cited on page 136.)

[Mount 1999] David M. Mount, Nathan S. Netanyahu and Jacqueline Le Moigne. *Efficient algorithms for robust feature matching.* Pattern Recogn., vol. 32, no. 1, pages 17–38, January 1999. (Cited on page 22.)

[Myatt 2002] D. R. Myatt, P. H. S. Torr, S. J. Nasuto, J. M. Bishop and R. Craddock. *NAPSAC: high noise, high dimensional robust estimation.* In In BMVC02, pages 458–467, 2002. (Cited on page 20.)

[Myronenko 2010] Andriy Myronenko and Xubo Song. *Point set registration: Coherent point drift.* IEEE transactions on pattern analysis and machine intelligence, vol. 32, no. 12, pages 2262–2275, 2010. (Cited on page 23.)

[Naimark 2002] L. Naimark and E. Foxlin. *Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker.* Proceedings International Symposium on Mixed and Augmented Reality, pages 27–36, 2002. (Cited on pages xvii and 14.)

[Nakai 2006] Tomohiro Nakai, Koichi Kise and Masakazu Iwamura. *Use of Affine Invariants in Locally Likely Arrangement Hashing for Camera-based Document Image Retrieval.* In Proc. of the 7th International Conference on Document Analysis Systems, DAS'06, pages 541–552, 2006. (Cited on pages 11, 14, 21, 24 and 25.)

[Nguyen 2015] Thanh Nguyen, Gerhard Reitmayr and Dieter Schmalstieg. *Structural Modeling from Depth Images.* IEEE Transactions on Visualization and Computer Graphics, vol. 21, no. 11, pages 1230–1240, 2015. (Cited on pages 113 and 114.)

[Ni 2009] Kai Ni, Hailin Jin and Frank Dellaert. *GroupSAC: Efficient consensus in the presence of groupings.* In 2009 IEEE 12th International Conference on Computer Vision, pages 2193–2200. IEEE, 2009. (Cited on page 20.)

[Nistér 2008] David Nistér and Henrik Stewénius. *Linear Time Maximally Stable Extremal Regions.* In 10th European Conference on Computer Vision, ECCV, pages 183–196, 2008. (Cited on page 99.)

[Olson 1997] Clark F. Olson. *Efficient Pose Clustering Using a Randomized Algorithm.* Int. J. Comput. Vision, vol. 23, no. 2, pages 131–147, June 1997. (Cited on page 21.)

[Ouellet 2008] Jean-Nicolas Ouellet, Félix Rochette and Patrick Hébert. *Geometric Calibration of a Structured Light System using Circular Control Points.* In 3D Data Processing, Visualization and Transmission, 3DPVT, pages 183–190, 2008. (Cited on pages xxi, 94 and 95.)

[Ozuysal 2010] Mustafa Ozuysal, Michael Calonder, Vincent Lepetit and Pascal Fua. *Fast keypoint recognition using random ferns.* IEEE transactions on pattern analysis and machine intelligence, vol. 32, no. 3, pages 448–461, 2010. (Cited on page 17.)

[Pavlakos 2015] Georgios Pavlakos and Kostas Daniilidis. *Reconstruction of 3D Pose for Surfaces of Revolution from Range Data.* In 3D Vision (3DV), 2015 International Conference on, pages 648–656. IEEE, 2015. (Cited on page 115.)

[Pham 2014] The-Anh Pham, Mathieu Delalandre, Sabine Barrat and Jean-Yves Ramel. *Accurate junction detection and characterization in line-drawing images.* Pattern Recogn., vol. 47, no. 1, pages 282–295, January 2014. (Cited on page 88.)

[Pottmann 1999] Helmut Pottmann, Martin Peternell and Bahram Ravani. *An introduction to line geometry with applications.* Computer-Aided Design, vol. 31, no. 1, pages 3–16, 1999. (Cited on pages 115, 116, 121, 126, 127 and 128.)

[Qiu 2014] Rongqi Qiu, Qian-Yi Zhou and Ulrich Neumann. *Pipe-Run Extraction and Reconstruction from Point Clouds.* In 13th European Conference on Computer Vision, ECCV, pages 17–30. Springer, 2014. (Cited on page 114.)

[Raguram 2013a] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas and Jan-Michael Frahm. *USAC: A Universal Framework for Random Sample Consensus.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pages 2022–2038, August 2013. (Cited on pages 20 and 83.)

[Raguram 2013b] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jirí Matas and Jan-Michael Frahm. *USAC: A Universal Framework for Random Sample Consensus.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pages 2022–2038, August 2013. (Cited on pages 43 and 44.)

[Reinbacher 2010] Christian Reinbacher, Matthias Ruther and Horst Bischof. *Pose estimation of known objects by efficient silhouette matching.* In Pattern Recognition (ICPR), 2010 20th International Conference on, pages 1080–1083. IEEE, 2010. (Cited on page 18.)

[Rekimoto 1998] J. Rekimoto. *Matrix: a realtime object identification and registration method for augmented reality.* In Computer Human Interaction, 1998. Proceedings. 3rd Asia Pacific, pages 63–68, Jul 1998. (Cited on pages v and 14.)

[Resch 2015] Christoph Resch, Hemal Naik, Peter Keitler, Steven Benkhardt and Gudrun Klinker. *On-site Semi-Automatic Calibration and Registration of a Projector-Camera System Using Arbitrary Objects With Known Geometry.* IEEE Transactions on Visualization and Computer Graphics, vol. 21, no. 11, pages 1211–1220, 2015. (Cited on pages xxi, 92 and 93.)

[Ridden 2013] Paul Ridden. *IKEA catalog uses augmented reality to give a virtual preview of furniture in a room.* http://newatlas.com/ikea-augmented-reality-catalog-app/28703/, 2013. [Online; accessed 02-August-2016]. (Cited on pages xvii and 3.)

[Rigoutsos 1995] Isidore Rigoutsos and Robert Hummel. *A Bayesian approach to model matching with geometric hashing.* Computer vision and image understanding, vol. 62, no. 1, pages 11–26, 1995. (Cited on page 21.)

[Rosenfeld 1968] Azriel Rosenfeld and John L Pfaltz. *Distance functions on digital pictures.* Pattern recognition, vol. 1, no. 1, pages 33–61, 1968. (Cited on page 18.)

[Rosten 2006] Edward Rosten and Tom Drummond. *Machine Learning for High-speed Corner Detection.* In Proceedings of the 9th European Conference on Computer Vision - Volume Part I, ECCV'06, pages 430–443, Berlin, Heidelberg, 2006. Springer-Verlag. (Cited on page 15.)

[Rosten 2010] Edward Rosten, Reid Porter and Tom Drummond. *Faster and better: A machine learning approach to corner detection.* IEEE transactions on

pattern analysis and machine intelligence, vol. 32, no. 1, pages 105–119, 2010. (Cited on page 15.)

[Rothwell 1992] Charles A Rothwell, Andrew Zisserman, David A Forsyth and Joseph L Mundy. *Canonical frames for planar object recognition.* In European Conference on Computer Vision, pages 757–772. Springer, 1992. (Cited on page 18.)

[Rublee 2011] Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary Bradski. *ORB: An Efficient Alternative to SIFT or SURF.* In Proc. of the 2011 IEEE International Conference on Computer Vision, ICCV '11, pages 2564–2571, 2011. (Cited on page 83.)

[Schilling 2012] Mark F Schilling. *The Surprising Predictability of Long Runs.* Math. Mag., vol. 85, no. 2, pages 141–149, April 2012. (Cited on page 72.)

[Schnabel 2007] Ruwen Schnabel, Roland Wahl and Reinhard Klein. *Efficient RANSAC for Point-Cloud Shape Detection.* Computer Graphics Forum, vol. 26, no. 2, pages 214–226, June 2007. (Cited on page 114.)

[Shapiro 1992] Larry S Shapiro and J Michael Brady. *Feature-based correspondence: an eigenvector approach.* Image and vision computing, vol. 10, no. 5, pages 283–288, 1992. (Cited on page 24.)

[Shi 1994] Jianbo Shi and Carlo Tomasi. *Good Features to Track.* In Proc. of the 1994 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '94, pages 593–600, June 1994. (Cited on pages 15 and 84.)

[Shi 2008] Yonggang Shi and William Clement Karl. *A real-time algorithm for the approximation of level-set-based curve evolution.* Image Processing, IEEE Transactions on, vol. 17, no. 5, pages 645–656, 2008. (Cited on page 53.)

[Shih 1996] Sheng-Wen Shih, Yi-Ping Hung and Wei-Song Lin. *Accuracy Analysis on the Estimation of Camera Parameters for Active Vision Systems.* In 13th International Conference on Pattern Recognition, volume 1 of *ICPR*, pages 930–935, 1996. (Cited on pages 97 and 104.)

[Song 2015] Meng Song and Daniel Huber. *Automatic Recovery of Networks of Thin Structures.* In 3D Vision (3DV), 2015 International Conference on, pages 37–45. IEEE, 2015. (Cited on page 114.)

[Taguchi 2015] Yuichi Taguchi and Srikumar Ramalingam. *Method for fitting primitive shapes to 3D point clouds using distance fields*, December 2015. US Patent 9,208,609. (Cited on page 114.)

[Tang 2014] Jun Tang, Ling Shao and Xiantong Zhen. *Robust point pattern matching based on spectral context.* Pattern Recognition, vol. 47, no. 3, pages 1469–1484, 2014. (Cited on page 24.)

[Tateno 2015] Keisuke Tateno, Federico Tombari and Nassir Navab. *Real-time and scalable incremental segmentation on dense SLAM.* In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 4465–4472. IEEE, 2015. (Cited on pages 125, 126 and 129.)

[Tateno 2016] Keisuke Tateno, Federico Tombari and Nassir Navab. *When 2.5D is not enough: Simultaneous Reconstruction, Segmentation and Recognition on dense SLAM.* In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 2295–2302. IEEE, 2016. (Cited on page 114.)

[Tombari 2013a] Federico Tombari, Alessandro Franchi and Luigi Di Stefano. *Bold features to detect texture-less objects.* In Proceedings of the IEEE International Conference on Computer Vision, pages 1265–1272, 2013. (Cited on page 19.)

[Tombari 2013b] Federico Tombari, Samuele Salti and Luigi Di Stefano. *Performance Evaluation of 3D Keypoint Detectors.* Int. J. Comput. Vision, vol. 102, no. 1-3, pages 198–220, March 2013. (Cited on page 80.)

[Tsin 2004] Yanghai Tsin and Takeo Kanade. *A Correlation-Based Approach to Robust Point Set Registration.* In Proc. of the 8th European Conference on Computer Vision, ECCV 2004, pages 558–569, 2004. (Cited on page 23.)

[Uchiyama 2009] Hideaki Uchiyama, Hideo Saito, Myriam Servières and Guillaume Moreau. *AR GIS on a Physical Map based on Map Image Retrieval using LLAH Tracking.* In IAPR Conference on Machine Vision Application, Yokohama, Japon, 2009. (Cited on page 26.)

[Uchiyama 2011a] Hideaki Uchiyama and E. Marchand. *Toward Augmenting Everything: Detecting and Tracking Geometrical Features on Planar Objects.* In Mixed and Augmented Reality (ISMAR), 10th IEEE International Symposium on, pages 17–25, 2011. (Cited on pages 84 and 85.)

[Uchiyama 2011b] Hideaki Uchiyama and Hideo Saito. *Random Dot Markers.* In Proc. of the 2011 IEEE Virtual Reality Conference, VR '11, pages 271–272, 2011. (Cited on pages vi, xvii, 10, 14, 24, 25, 44, 76 and 88.)

[Uchiyama 2011c] Hideaki Uchiyama, Hideo Saito, Myriam Servières and Guillaume Moreau. *Camera tracking by online learning of keypoint arrangements using LLAH in augmented reality applications.* Virtual Real., vol. 15, no. 2-3, pages 109–117, June 2011. (Cited on pages vi and 10.)

[Uchiyama 2012] Hideaki Uchiyama and Eric Marchand. *Object detection and pose tracking for augmented reality: Recent approaches.* In 18th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV), 2012. (Cited on page 6.)

[Umeyama 1991] Shinji Umeyama. *Least-Squares Estimation of Transformation Parameters Between Two Point Patterns.* IEEE Trans. Pattern Anal. Mach. Intell., vol. 13, no. 4, pages 376–380, April 1991. (Cited on page 75.)

[Van Nguyen 2013] Hien Van Nguyen and Fatih Porikli. *Support vector shape: A classifier-based shape representation.* IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 4, pages 970–982, 2013. (Cited on pages 18 and 137.)

[Van Wamelen 2004] Paul B Van Wamelen, Z Li and SS Iyengar. *A fast expected time algorithm for the 2-D point pattern matching problem.* Pattern Recognition, vol. 37, no. 8, pages 1699–1711, 2004. (Cited on page 21.)

[Wang 2009a] Lu Wang, Ulrich Neumann and Suya You. *Wide-baseline image matching using line signatures.* In 2009 IEEE 12th International Conference on Computer Vision, pages 1311–1318. IEEE, 2009. (Cited on pages 15 and 19.)

[Wang 2009b] Zhiheng Wang, Fuchao Wu and Zhanyi Hu. *MSLD: A robust descriptor for line matching.* Pattern Recognition, vol. 42, no. 5, pages 941–953, 2009. (Cited on page 16.)

[Wang 2012] Xiaoyun Wang and Xianquan Zhang. *Point pattern matching algorithm for planar point sets under Euclidean transform.* Journal of Applied Mathematics, vol. 2012, 2012. (Cited on page 24.)

[Wang 2013] Ke Wang, Tielin Shi, Guanglan Liao and Qi Xia. *Image registration using a point-line duality based line matching method.* Journal of Visual

Communication and Image Representation, vol. 24, no. 5, pages 615–626, 2013. (Cited on page 19.)

[Willis 2003] Andrew Willis, Xavier Orriols and David B Cooper. *Accurately Estimating Sherd 3D Surface Geometry with Application to Pot Reconstruction.* In Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on, volume 1, pages 5–5. IEEE, 2003. (Cited on page 115.)

[Wolfson 1997] Haim J. Wolfson and Isidore Rigoutsos. *Geometric Hashing: An Overview.* IEEE Comput. Sci. Eng., vol. 4, no. 4, pages 10–21, October 1997. (Cited on pages 11, 21, 65 and 76.)

[Yamauchi 2008] Koichiro Yamauchi, Hideo Saito and Yukio Sato. *Calibration of a Structured Light System by Observing Planar Object from Unknown Viewpoints.* In 19th International Conference on Pattern Recognition, ICPR, pages 1–4, 2008. (Cited on pages xxi and 92.)

[Yamazaki 2011] Shuntaro Yamazaki, Masaaki Mochimaru and Takeo Kanade. *Simultaneous self-calibration of a projector and a camera using structured light.* In IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 60–67, 2011. (Cited on pages 92 and 93.)

[Yin 2006] Peng-Yeng Yin. *Particle swarm optimization for point pattern matching.* Journal of Visual Communication and Image Representation, vol. 17, no. 1, pages 143–162, 2006. (Cited on page 24.)

[Yin 2012] Peng-Yeng Yin. *Scatter search for point pattern matching: A comparative study.* In Natural Computation (ICNC), 2012 Eighth International Conference on, pages 1084–1088. IEEE, 2012. (Cited on page 24.)

[Zhang 2000] Zhengyou Zhang. *A Flexible New Technique for Camera Calibration.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pages 1330–1334, 2000. (Cited on pages 93, 94, 95, 96 and 104.)

[Zhang 2003] Lihua Zhang, Wenli Xu and Cheng Chang. *Genetic algorithm for affine point pattern matching.* Pattern Recognition Letters, vol. 24, no. 1, pages 9–19, 2003. (Cited on page 24.)

[Zhang 2006] Song Zhang and Peisen S Huang. *Novel method for structured light system calibration.* Optical Engineering, vol. 45, no. 8, pages 083601–083601, 2006. (Cited on page 93.)

[Zhang 2013] Lilian Zhang and Reinhard Koch. *An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency.* Journal of Visual Communication and Image Representation, vol. 24, no. 7, pages 794–805, 2013. (Cited on page 15.)

[Zhang 2014] Zhengyou Zhang. Camera calibration. Springer, 2014. (Cited on page 6.)

[Zhong 2009] Yu Zhong. *Intrinsic Shape Signatures: A Shape Descriptor for 3D Object Recognition.* In Proc. of the IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), pages 689–696, 2009. (Cited on page 80.)

[Zhou 2008] Feng Zhou, Henry Been-Lirn Duh and Mark Billinghurst. *Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR.* In Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality, pages 193–202. IEEE Computer Society, 2008. (Cited on page 13.)

UNIVERSITE BRETAGNE LOIRE

Centrale Nantes

Thèse de Doctorat

Liming YANG

Titre de la thèse : Recalage robuste à base de motifs de points pseudo aléatoires pour la réalité augmentée

Title of thesis: Point pattern matching for augmented reality

Résumé

La Réalité Augmentée (RA) vise à afficher des informations numériques virtuelles sur des images réelles. Le recalage est important, puisqu'il permet d'aligner correctement les objets virtuels dans le monde réel. Contrairement au tracking qui recale en utilisant les informations de l'image précédente, la localisation à grande échelle (*wide baseline localization*) calcule la solution en utilisant uniquement les informations présentes dans l'image courante. Il permet ainsi de trouver des solutions initiales au problème de recalage (*initialisation*) et, n'est pas sujet aux problèmes de « perte de tracking ». Le problème du recalage en RA est relativement bien étudié dans la littérature, mais les méthodes existantes fonctionnent principalement lorsque la scène augmentée présente des textures. Pourtant, pour le recalage avec les objets peu ou pas texturés, il est possible d'utiliser leurs informations géométriques qui représentent des caractéristiques plus stables que les textures.

Cette thèse s'attache au problème de recalage basé sur des informations géométriques, et plus précisément sur les points. Nous proposons deux nouvelles méthodes de recalage de points (RRDM et LGC) robustes et rapides. LGC est une amélioration de la méthode RRDM et peut mettre en correspondance des ensembles de motifs de points 2D ou 3D subissant une transformation dont le type est connu. LGC présente un comportement linéaire en fonction du nombre de points, ce qui permet un tracking en temps-réel. La pertinence de LGC a été illustrée en développant une application de calibration de système projecteur-caméra dont les résultats sont comparables avec l'état de l'art tout en présentant des avantages pour l'utilisateur en termes de taille de mire de calibration..

Mots-clés
réalité augmentée, recalage des motifs de points, suivi, peu-texturé, calibration, projecteur-caméra, carte augmentée, surface de révolution

Abstract

Registration is a very important task in Augmented Reality (AR). It provides the spatial alignment between the real environment and virtual objects. Unlike tracking (which relies on previous frame information), wide baseline localization finds the correct solution from a wide search space, so as to overcome the initialization or tracking failure problems. Nowadays, various wide baseline localization methods have been applied successfully. But for objects with no or little texture, there is still no promising method. One possible solution is to rely on the geometric information, which sometimes does not vary as much as texture or color.

This dissertation focuses on new wide baseline localization methods entirely based on geometric information, and more specifically on points. I propose two novel point pattern matching algorithms, RRDM and LGC. Especially, LGC registers 2D or 3D point patterns under any known transformation type and supports multi-pattern recognitions. It has a linear behavior with respect to the number of points, which allows for real-time tracking. It is applied to multi targets tracking and augmentation, as well as to 3D model registration. A practical method for projector-camera system calibration based on LGC is also proposed. It can be useful for large scale Spatial Augmented Reality (SAR). Besides, I also developed a method to estimate the rotation axis of surface of revolution quickly and precisely on 3D data. It is integrated in a novel framework to reconstruct the surface of revolution on dense SLAM in real-time.

Key Words
augmented reality, point pattern matching, registration, texture-less tracking, calibration, projector-camera, augmented map, surface of revolution